DIEGO JONATHAN HOSS

DASFLOW: UMA ARQUITETURA DISTRIBUÍDA DE ARMAZENAMENTO E PROCESSAMENTO PARA DADOS DE MONITORAMENTO DE REDE

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientadora: Prof[a]. Dr[a]. Carmem Satie Hara

CURITIBA

2015

H829d Hoss, Diego Jonathan
Dasflow : uma arquitetura distribuída de armazenamento e processamento para dados de monitoramento de rede/ Diego Jonathan Hoss. – Curitiba, 2015.
92 f. : il. color. ; 30 cm.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-graduação em Informática, 2015.

Orientador: Carmem Satie Hara .
Bibliografia: p. 88-92.

1. Redes de computação - Administração. 2. Processamento eletrônico de dados - Processamento distribuído. 3. Metadados. I. Universidade Federal do Paraná. II.Hara, Carmem Satie. III. Título.

CDD: 004.36

Ministério da Educação
Universidade Federal do Paraná
Programa de Pós-Graduação em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno Diego Hoss, avaliamos o trabalho intitulado, "DASFlow: Uma Arquitetura Distribuída de Armazenamento e Processamento de Dados de Monitoramento de Rede", cuja defesa foi realizada no dia 24 de setembro de 2015, às 14:00 horas, no Departamento de Informática do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela:
(X) **aprovação** do candidato. ( ) **reprovação** do candidato.

Curitiba, 24 de setembro de 2015.

Carmem Satie Hara
Profª. Dra. Carmem Satie Hara
PPGInf - Orientadora

Prof. Dr. Mario Dantas
UFSC – Membro Externo

Prof. Dr. Elias Procópio Duarte Júnior
PPGInf – Membro Interno

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

AGRADECIMENTOS

Agradeço primeiramente a minha família. Em especial ao meu pai Romualdo e minha mãe Salete por me ensinar a ser a pessoa que sou hoje, em todos os sentidos. Não existem palavras que descrevam a imensa gratidão que sinto por vocês. Ao meu irmão Douglas, que por meio do exemplo, me faz acreditar que sempre posso fazer algo a mais.

Agradeço a minha companheira Roberta. Sempre de braços abertos para se entristecer e se alegrar ao meu lado. De todas as experiências boas e ruins que compartilhou comigo, nenhuma foi tão significativa quanto o seu próprio exemplo. Aos seus pais, familiares e amigos, que de uma forma ou de outra compartilharam desta luta e me ajudaram a superar os desafios.

Meus sinceros e profundos agradecimentos à professora Carmem Hara. Com sua persistência, perseverança e principalmente paciência, conseguiu fazer de um galho torto e cheio de nós, um lápis, que muito embora ainda sem jeito, seja capaz de escrever algumas palavras. Aos colegas de laboratório, que na medida do possível, sempre me ajudaram e tornaram as coisas menos difíceis. Para não esquecer de ninguém não vou citá-los, porém, sintam-se todos agradecidos, profundamente.

Agradeço aos professoras Elias P. Duarte Jr. e Mário Dantas, que ao aceitarem participar da banca, deram contribuições muito importantes fortalecendo meu aprendizado, em todos os aspectos. Agradeço ao PoP-PR, representado na pessoa do Christian Lyra, o qual considero meu coorientador, pois sempre esteve à disposição para me orientar e ajudar quando precisei.

Aos demais professores e secretaria do departamento, colegas de trabalho e amigos que aqui não foram citados mas contribuíram no que lhes era possível, o meu sincero agradecimento.

E por fim, o agradecimento mais especial, à Deus. Todas as realizações, todas as pessoas que comigo caminharam nesta jornada, só o fizeram porque Deus estava ao nosso lado, caminhando conosco. Sem ele, nada disso teria sido possível.

"Ainda que eu andasse pelo vale da sombra da morte, não temeria mal algum, porque tu estás comigo."

Salmo 23.4

RESUMO

O monitoramento de redes é uma atividade pertencente à área de gerência de redes, na qual efetua-se a coleta, armazenamento, processamento e análise dos dados de monitoramento. Para realizar esta atividade, existem ferramentas que implementam tais funcionalidades. Grande parte destas ferramentas são baseadas na arquitetura centralizada, e entre elas encontra-se o NfSen/Nfdump. Esta ferramenta é amplamente utilizada pelos administradores de rede por possuir boa documentação e ser de código aberto. O modelo centralizado possui limitações associadas à escalabilidade que são inerentes à arquitetura. Entre elas está a falta de redundância, um ponto único de falha e a ausência de balanceamento de carga. Isto significa que as ferramentas com arquitetura centralizada estão sujeitas a essas limitações, ou seja, existem limites no volume de armazenamento bem como na sua capacidade de coleta e processamento. Na literatura, encontram-se soluções para este problema baseadas em compressão dos dados e armazenamento distribuído de dados. Nesta dissertação, é proposta uma arquitetura distribuída chamada **DASFlow** aplicada à ferramenta de monitoramento de rede NfSen/Nfdump, cujo o objetivo é prover escalabilidade de coleta, armazenamento e processamento. Para isso, a arquitetura define os módulos StoreDAS-Cliente e StoreDAS-Servidor que atuam em conjunto com o sistema de arquivos distribuído (SAD) para prover escalabilidade de coleta e armazenamento. A escalabilidade de processamento é fornecida pelos módulos QueryDAS-Cliente e QueryDAS-Servidor. A arquitetura também prevê a existência do módulo de Metadados responsável por manter as informações sobre o armazenamento e distribuição dos dados de monitoramento. Os resultados experimentais mostram o potencial da arquitetura proposta. O DASFlow obteve menores tempos de resposta para o processamento das consultas mais frequentes que variam entre 13% e 34%, se comparados à ferramenta NfSen/Nfdump. Adicionalmente, a adoção de um sistema de arquivos distribuído mostra-se eficaz ao prover escalabilidade para o armazenamento dos dados de monitoramento de rede.

ABSTRACT

Network monitoring is one of the activities of the network management field, in which one collects, stores, processes and analyzes monitoring data. It relies on tools that implement such functionalities. Many of these tools are based on a centralized architecture, and among them is NfSen/Nfdump. NfSen/Nfdump is widely used among network administrators, due to a good documentation and the fact that it is open source. The centralized model has some limitations with respect to scalability, which are inherent to the architecture. One of them is the lack of redundancy, single point of failure and the absence of load balancing. As a result, centralized architecture tools are subject to the some limitations. That is, storage capacity is limited as well as the ability to collect and process data. Solutions to solve these problems, based on data compression and distributed data storage, can be found in the literature. In this dissertation, we propose a distributed architecture called DASFlow, which provides scalability for data collection, storage and processing. In this regard, the architecture defines the StoreDAS-Cliente and StoreDAS-Servidor modules, which work together with a distributed file system (DFS) in order to provide data collection and storage scalability. Processing scalability is provided by the QueryDAS-Cliente and QueryDAS-Servidor modules. The architecture also contains a Metadata module, responsible for keeping the information about storage and distribution of monitoring data. The architecture has been implemented with the NfSen/Nfdump network monitoring tool and Ceph distributed file system. The experimental results show that the DASFlow architecture has achieved shorter response times for the processing of the most frequent queries, which vary between 13% and 34%, if compared to the original NfSen/Nfdump tool. Additionally, the use of a distributed file system has proved to be efficient in providing scalability for the storage of network monitoring data.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

A contínua pervasividade da tecnologia nos diversos setores da sociedade acarretou em uma mudança de paradigma no modo como é feita a comunicação entre pessoas, empresas e máquinas. Essa nova era trouxe um aumento exponencial no volume de dados que trafega diariamente na Internet [32]. Por conta desse acréscimo e também da evolução desordenada da tecnologia, as redes de comunicação tornaram-se cada vez mais complexas em sua estrutura. Desta forma, monitorar e gerenciar grandes redes tornou-se uma tarefa desafiadora, especialmente pela dificuldade em lidar com enormes volumes de dados.

No contexto de gerência de redes, algumas atividades preliminares são a base para as etapas posteriores. Entre elas, está a fase de monitoramento do tráfego de rede, ou simplesmente, monitoramento de rede. Esta etapa é responsável por efetuar a coleta, armazenamento, recuperação e apresentação dos dados de monitoramento, conhecidos também como fluxos IP (*Internet Protocol*). O monitoramento de rede é realizado utilizando-se diversas ferramentas conhecidas como ferramentas de monitoramento de rede. Grande parte destas ferramentas possuem limitações em relação à manipulação de grandes massas de dados. Entre as dificuldades, está a arquitetura com a qual efetuam a coleta, armazenamento e processamento dos fluxos IP, que em geral é centralizada. O modelo centralizado possui limitações inerentes à arquitetura, podendo ser destacadas a falta de redundância, um ponto único de falha e a ausência de balanceamento de carga.

Atualmente, trabalhos encontrados na literatura buscam prover escalabilidade para a atividade de monitoramento de rede e contornar o problema da arquitetura centralizada [13, 24]. A escalabilidade para o monitoramento é descrita como a capacidade de realizar as tarefas de coleta, armazenamento e recuperação sem limites nas quantidades de dados manipulados. Para isto, os trabalhos apresentam soluções com a utilização de ambientes distribuídos [10, 21], sendo que a maioria dos trabalhos foca na escalabilidade de um

aspecto do monitoramento, como processamento e recuperação [27], armazenamento [26] ou coleta [28]. Ainda que o ambiente de armazenamento distribuído permita tornar a atividade de monitoramento escalável, ele possui um custo na execução de consultas. Isto ocorre porque os dados são armazenados em diversos nós da rede. Desta forma, quando uma consulta é executada, é somado ao seu tempo de resposta o custo da latência de rede para agrupar os dados necessários para atender a requisição. A fim de diminuir as trocas de mensagens na rede e otimizar o desempenho das consultas, os dados que são frequentemente acessados devem ser agrupados e armazenados de modo organizado nos servidores de armazenamento. Para tornar isto possível, é necessário obter o controle de localidade dos dados armazenados [35].

Os ambientes distribuídos presentes na literatura podem ser divididos em duas abordagens principais. A abordagem mais utilizada é por meio de redes *Peer-to-Peer* (P2P) e DHT (*Distributed Hash Table*). Esta técnica é muito eficiente no processo de escrita porque utiliza um par (*chave*, *valor*) para encapsular os dados e armazená-los no ambiente. Esta abordagem no entanto, não fornece mecanismos para que a aplicação determine a localidade de um dado armazenado. Isto é, uma aplicação que utiliza DHT não pode indicar com precisão em qual servidor do ambiente o dado será armazenado. A outra abordagem utiliza sistemas de arquivos distribuídos para criar o ambiente distribuído. Este modelo difere-se daqueles baseados em DHT porque em geral estes sistemas oferecem mecanismos para que a aplicação possa controlar a localidade dos dados armazenados.

No contexto de monitoramento de redes existem diversas ferramentas utilizadas para este fim. Cada ferramenta possui suas vantagens e desvantagens. Algumas ferramentas, como o Tcpdump [39] e Wireshark [44], manipulam pacotes IP. Em termos de monitoramento, os pacotes IP são considerados dados brutos. Eles carregam o tráfego de rede entre destinos e origens sem realizar medições sobre as informações transmitidas. Isto é uma vantagem quando se deseja obter, de modo mais técnico, informações sobre a rede monitorada. Em contrapartida, ferramentas como Flow-tools [12] e Ntop [8] tratam de fluxos IP. Esse tipo de dado de monitoramento possui como vantagem ser um dado manipulado e conter informações contabilizadas sobre o tráfego de rede, como por exemplo:

soma de *bytes* trafegados e porcentagem de utilização de uma rede.

Uma das ferramentas utilizadas para a atividade de monitoramento e que também manipula fluxos IP é o NfSen/Nfdump [30]. Esta ferramenta realiza as funções de coleta, armazenamento, processamento e apresentação das informações sobre o tráfego de rede. Sua interface gráfica fornece um ambiente intuitivo e de fácil utilização. Por meio de consultas pré-existentes, ela é capaz de separar e apresentar os resultados em forma de gráficos para diferentes objetivos. A ferramenta é muito utilizada em diversas instituições por ser gratuita e possuir ampla documentação. Embora seja bem aceita pelos administradores de rede, o NfSen/Nfdump possui arquitetura centralizada e está sujeita, portanto, às limitações deste modelo. Neste sentido, este trabalho busca fornecer mecanismos para tornar o NfSen/Nfdump uma ferramenta escalável, permitindo desta forma, que ela seja capaz de acompanhar o crescente aumento no tráfego de rede.

1.1 Objetivo Geral

O objetivo deste trabalho é propor uma solução capaz de prover escalabilidade para a atividade de monitoramento de rede. Para isto, criou-se uma arquitetura distribuída chamada **DASFlow** aplicada à ferramenta NfSen/Nfdump. Ela é composta de um conjunto de servidores de armazenamento, de forma que o processamento sobre estes dados possa ser executado em paralelo. O funcionamento da arquitetura é baseado no desenvolvimento de três módulos, chamados **Metadados, StoreDAS e QueryDAS**, que atuam em todos servidores, permitindo sua integração e cooperação para a execução das tarefas. O armazenamento dos dados de monitoramento é realizado utilizando um sistema de arquivos distribuído (SAD).

Desta forma, é esperado que a arquitetura proposta neste trabalho contemple os três aspectos do monitoramento (coleta, armazenamento e processamento). Embora a arquitetura não esteja completamente implementada, os resultados mostram um potencial da arquitetura como um todo. Nota-se que além do aumento na capacidade de armazenamento provido pelo SAD, a arquitetura obteve ganhos no tempo de resposta para o processamento das consultas mais utilizadas. O ganho obtido neste trabalho variou entre

13% e 34% se comparado à ferramenta NfSen/Nfdump em seu formato original.

1.2 Objetivos Específicos

Para alcançar o objetivo geral deste trabalho, considera-se as atividades específicas descritas a seguir.

- Compreender o funcionamento da ferramenta de monitoramento de rede NfSen/Nfdump;
- Comparar os sistemas de arquivos distribuídos mais utilizados atualmente;
- Planejar a arquitetura DASFlow com seus componentes e funcionamento;
- Especificar os módulos que compõem a arquitetura: StoreDAS, QueryDAS e Metadados;
- Implementar e avaliar o funcionamento dos módulos Metadados, QueryDAS-Cliente e QueryDAS-Servidor;

1.3 Escopo do Trabalho

Para este trabalho, considera-se a implementação e avaliação dos módulos Metadados, QueryDAS-Cliente e QueryDAS-Servidor. Isto significa que embora a arquitetura DASFlow contemple os três aspectos da escalabilidade, apenas os aspectos de processamento e armazenamento são analisados nesta versão da implementação da arquitetura. Assim, a implementação dos módulos StoreDAS-Cliente e StoreDAS-Servidor, provedores da escalabilidade de coleta, são tratados como trabalhos futuros.

1.4 Organização do Trabalho

O restante deste trabalho está organizado da seguinte maneira. O capítulo 2 traz conceitos sobre o monitoramento de rede, as ferramentas mais utilizadas neste processo e finaliza com uma análise comparativa. O capítulo 3 apresenta uma discussão sobre os

trabalhos encontrados na literatura que tratam da arquitetura distribuída para o monitoramento de rede. No capítulo 4 são descritos os principais conceitos relacionados aos sistemas de arquivos distribuídos, seguido de uma apresentação e análise dos sistemas mais conhecidos. Em seguida, o capítulo 5 apresenta a arquitetura DASFlow de forma detalhada, descrevendo seus componentes e funcionamento. O capítulo 6 traz os experimentos e resultados obtidos a partir da arquitetura proposta com uma discussão sobre as vantagens e desvantagens da solução. O capítulo 7 finaliza o trabalho apresentando a conclusão seguida dos desafios e melhorias como objetivos futuros para este trabalho.

CAPÍTULO 2

MONITORAMENTO DE REDES

O monitoramento de redes é uma das tarefas relacionadas à gerência de redes. Este processo é composto de quatro atividades que são: coleta, armazenamento, processamento e apresentação do tráfego monitorado [2]. As demais tarefas associadas à gerência de redes baseiam-se nos dados coletados e armazenados durante a etapa de monitoramento.

A tarefa de monitoramento pode ser realizada de maneira simples e diretamente sobre o tráfego de rede, ou, de modo elaborado, efetuando o monitoramento de serviços e aplicações de rede utilizando o protocolo SNMP (*Simple Network Management Protocol*) [5]. Para realizar o monitoramento de serviços e aplicações geralmente utilizam-se ferramentas que implementam o protocolo SNMP, entre elas tem-se o OpenNMS [16], Zabbix [25] e Nagios [29]. O monitoramento do tráfego é realizado de forma simples atuando apenas sobre os dados que trafegam na rede, sendo esta a forma de monitoramento tratada no presente trabalho.

Muitos são os motivos que levam as organizações a monitorarem seu tráfego de rede. As operadoras de telecomunicações, por exemplo, usam a fase de monitoramento para obter informações a fim de realizar a cobrança pelo *link* comercializado. Ainda por meio desta etapa, elas otimizam a utilização da banda com técnicas de qualidade de serviço (do inglês, *Quality of Service - QoS*). Questões relacionadas ao desempenho e segurança também são consideradas. Com o correto monitoramento do tráfego, pode-se entender o comportamento da rede e prevenir, ou minimizar, problemas de ataques de negação de serviço (do inglês, *Denial of Service - DoS*), ou outros mais sofisticados, como os ataques distribuídos de negação de serviço (*Distributed Denial of Service - DDoS*).

Além das exigências técnicas, existem leis e diretivas na União Européia [31], e recentemente no Brasil com o Marco Civil da Internet [11], que obrigam as operadoras de telecomunicações a monitorarem e reterem os dados de tráfego por um longo período. Isto

é determinante para possibilitar buscas futuras relacionadas a crimes, processos judiciais, defesa nacional, entre outros.

As operadoras são responsáveis por concentrar uma grande parte do tráfego da Internet. Entretanto, as leis e obrigações técnicas envolvidas no processo de monitoramento são aplicadas a várias outras entidades, tais como: universidades, empresas, hospitais e departamentos do governo. O monitoramento nestes locais se faz necessário por essas instituições contribuírem significativamente para o aumento de tráfego, que cresce exponencialmente ano após ano [32].

Nas seções seguintes serão abordados os principais conceitos relacionados ao monitoramento, seguido de uma apresentação das ferramentas que auxiliam neste processo, finalizando com uma análise comparativa entre as ferramentas frente aos conceitos apresentados.

2.1 Arquitetura e Conceitos do Monitoramento de Redes

O monitoramento de rede é uma das atividades pertencentes ao contexto de gerência de redes. Ela é composta por um conjunto de etapas, dentre as quais destaca-se: coleta, armazenamento, processamento e apresentação dos dados monitorados. Para que possa ser realizado, um conjunto de elementos é utilizado, sendo cada componente responsável por uma ou mais etapas. O agrupamento destes elementos compõe uma arquitetura de monitoramento. A Figura 2.1 ilustra os componentes de um esquema tradicional de monitoramento de rede.

Baseado na Figura 2.1, nesta seção são apresentados alguns conceitos e definições dos principais elementos dessa arquitetura.

2.1.1 Exportadores

Os elementos exportadores representam os dispositivos de rede conhecidos como roteadores ou *switches* de camada 3 (modelo OSI [38]). Eles são responsáveis por permitir a comunicação entre redes distintas. Na sua maioria, os equipamentos de interligação de

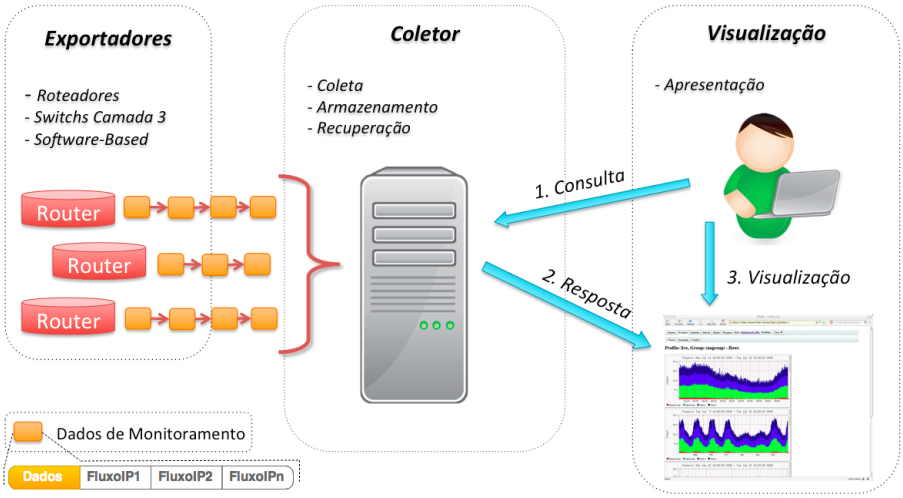

Figura 2.1: Arquitetura tradicional de um ambiente de monitoramento.

redes possuem recursos para contabilizar o tráfego que passa por eles. Desta forma, estes dispositivos podem ser configurados para gerar fluxos IP. Os fluxos IP são empacotados dando origem aos dados de monitoramento. Os dados de monitoramento possuem um formato bem definido geralmente determinado por um protocolo, dentre os quais podem ser citados: NetFlow [3], IPFIX [4] e sFlow [34]. Após serem gerados, os dados de monitoramento são enviados para outro elemento dessa arquitetura denominado coletor. Além dos dispositivos de rede, há também ferramentas que atuam como geradores e exportadores de dados de monitoramento, como por exemplo o nProbe [7]. Sua função consiste em receber todo o tráfego que passa por um roteador, transformá-lo em fluxos IP, empacotá-los e enviá-los para um coletor.

2.1.2 Dados de monitoramento

Geralmente contendo um conjunto de fluxos IP (do inglês, *IP flow*), os dados de monitoramento são gerados a partir dos roteadores. Um fluxo IP contém informações contabilizadas sobre uma comunicação de rede unidirecional ou bidirecional, entre uma origem e um destino em um intervalo de tempo [3, 4, 34]. A estrutura de um fluxo IP possui variações de acordo com o protocolo utilizado. Todos os protocolos no entanto, possuem ao menos as seguintes características: IP origem, IP destino, porta origem, porta destino, protocolo de comunicação (TCP, UDP, ICMP) e quantidade de dados trafegados (em *bytes*).

Dentre os protocolos mais conhecidos está o Netflow [3]. Ele é um protocolo para dados de monitoramento desenvolvido pela Cisco Systems[1]. Inicialmente projetado para atender os dispositivos de rede de sua fabricação, atualmente ele é um protocolo aberto. Ele possui várias versões sendo o Netflow versão 5 a mais utilizada. Além deste, outro protocolo existente e amplamente conhecido é o IPFIX [4]. Sua principal diferença está na personalização dos dados de monitoramento. Ele permite que características referentes ao tráfego monitorado sejam adicionadas ou removidas dos pacotes de fluxos IP. Outro protocolo também muito conhecido e utilizado é o sFlow [34]. Ele é um protocolo que foi concebido para ser implementado por qualquer marca de roteador. Este se difere dos demais porque sua contabilização de tráfego se dá estatisticamente utilizando amostragem, isto é, o monitoramento da rede feito utilizando este protocolo representa um comportamento aproximado. A Figura 2.2 mostra um exemplo simplificado de um fluxo IP

Fluxo IP						
IP Origem	**IP Destino**	**Porta Origem**	**Porta Destino**	**Protocolo**	**Bytes**	...
200.204.205.123	200.123.204.10	30041	80	TCP	41000	

Figura 2.2: Exemplo simplificado de um fluxo IP.

2.1.3 Coletor

No contexto de monitoramento de redes, algumas atividades são necessárias e compõem uma parte principal do processo, tais como: coleta, armazenamento e recuperação dos dados. Esta etapa é realizada pelo componente da arquitetura conhecido como coletor.

Assim como em qualquer outra atividade que requeira uma análise comportamental, o primeiro passo é a coleta. Existem basicamente duas formas de se efetuá-la: coleta por captura e coleta por recebimento. Quando é feita por captura, um coletor é colocado estrategicamente em um local da rede onde possa ficar "ouvindo" e capturando todo o tráfego. Neste modelo, os dados capturados são conhecidos como pacotes IP, e sob a

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

perspectiva de monitoramento, ainda são dados brutos de rede. Ao contrário dos fluxos IP, o dispositivo que captura é responsável por contabilizar e separar o tráfego por destinos e origens, protocolos, total em *bytes* de tráfego, etc. Em contrapartida, quando a coleta ocorre por meio de recebimento, o coletor fica aguardando que os exportadores enviem os fluxos IP já contabilizados.

Para que a coleta seja possível, é necessário que o coletor possua a capacidade de armazenar os dados. Desta forma, está presente na arquitetura um dispositivo de armazenamento. Geralmente, os dados de monitoramento são armazenados no próprio disco onde o coletor está sendo executado. Além desta, existem outras formas de armazenamento. Entre elas inclui-se um ambiente de armazenamento distribuído [26].

Tanto para a coleta quanto para o armazenamento dos dados, a arquitetura de monitoramento pode ser categorizada como sendo centralizada ou distribuída. A arquitetura mais comum e ilustrada na Figura 2.1 é a centralizada. Neste modelo, vários exportadores geram dados de monitoramento e os exportam para um único coletor. Na arquitetura distribuída, vários exportadores podem enviar os dados contendo os fluxos IP para vários coletores [27, 28].

2.1.4 Visualização

Nesta etapa, os coletores inicialmente recuperam os dados para em seguida apresentá-los. A apresentação dos dados de monitoramento pode ser feita de dois modos: na forma gráfica ou em modo texto. Quando um coletor que possui ambiente gráfico é acessado pelo usuário, ele disponibiliza de imediato telas e gráficos informativos [8, 30]. Isso acontece porque à medida em que os dados de monitoramento são armazenados, consultas pré-determinadas são aplicadas e filtram os dados para serem apresentados. Já na visualização em modo texto, um breve conhecimento técnico do utilizador é requerido. Isto ocorre porque ao contrário do modo gráfico, os dados de monitoramento não tiveram nenhum tratamento prévio. Quando o coletor é acessado, ele fica aguardando que consultas sejam aplicadas para que possa processar e retornar as informações solicitadas.

A fim de permitir a interação com o usuário, um coletor pode oferecer consultas com

filtros dinâmicos ou pré-definidos. Uma consulta com filtros pré-definidos é caracterizada por possuir um conjunto fechado de opções. Este modo restringe o grau de liberdade da consulta e é o mais comum em coletores com ambiente gráfico. Já nas consultas com filtros dinâmicos, o utilizador pode realizar a consulta definindo os parâmetros na forma e ordem em que forem necessários. Este é o mecanismo mais utilizado por coletores que operam apenas com modo texto, embora seja possível ambos os tipos de consulta em um mesmo coletor [30].

2.2 Ferramentas de Monitoramento de Redes

Na arquitetura de monitoramento de redes, nota-se que o coletor é responsável por todas as funções apresentadas anteriormente, ou seja, ele faz a coleta, armazenamento, processamento e apresentação dos dados. Sob a perspectiva do monitoramento do tráfego de rede, seja ele realizado sobre pacotes ou fluxos IP, existem algumas ferramentas que exercem a função de coletor, entre elas estão: Tcpdump [39], Wireshark [44], Ntop [8], Flow-tools [12] e NfSen/Nfdump [30], descritas a seguir.

2.2.1 Tcpdump

O Tcpdump [39] é uma ferramenta de código aberto muito utilizada como coletor de captura. Seu modo de operação consiste em ficar "ouvindo" o tráfego de uma rede que se deseja monitorar. Com isso, o Tcpdump captura todos os dados de rede na sua forma bruta, isto é, pacotes IP. Para isto, ele utiliza uma biblioteca escrita em C/C++ chamada LIBPCAP[2]. Após a captura, o Tcpdump armazena, de modo centralizado, todos os dados em um arquivo com extensão ".pcap". Com o intuito de disponibilizar as informações para o usuário, o Tcpdump permite que os dados sejam extraídos por consultas dinâmicas e apresenta-os em modo texto.

[2]http://sourceforge.net/projects/libpcap/

2.2.2 Wireshark

A ferramenta Wireshark [44] é muito semelhante ao Tcpdump em vários aspectos, um deles por ser de código aberto. No quesito captura de dados, esta ferramenta utiliza a mesma biblioteca (LIBPCAP) e trabalha com pacotes IP. Para o armazenamento conserva o mesmo padrão do Tcpdump, retendo todos os dados em um único arquivo e de modo centralizado. Para as consultas, o Wireshark permite consultas com filtros dinâmicos definidos pelo usuário. Embora possua muitas similaridades, o Wireshark pode ser considerado uma evolução em termos de visualização em relação ao Tcpdump, pois possui ambiente gráfico e permite alguns tipos de relatórios, gráficos e estatísticas.

2.2.3 Ntop

O Ntop [8] é atualmente conhecido também como Ntopng (Ntop *next generation*). Ao contrário das ferramentas apresentadas anteriormente, ele executa a captura por meio de recebimento e os dados de monitoramento são compostos de fluxos IP. Sua arquitetura de coleta e armazenamento padrão é centralizada. Em termos de consulta, ele fornece recursos que apresentam os resultados por meio de gráficos, relatórios e estatísticas. Esta ferramenta possui uma limitação relacionada à flexibilidade das consultas, pois fornece para o usuário apenas conjuntos pré-determinados de operações possíveis.

2.2.4 Flow-tools

O Flow-tools [12] é uma ferramenta de código aberto desenvolvida no formato de biblioteca. Esta ferramenta executa a coleta por meio de recebimento e permite processar, armazenar e apresentar dados de fluxo IP. A arquitetura de coleta e armazenamento é por padrão centralizada. O Flow-tools fornece consultas com filtros dinâmicos, definidos pelo usuário, e apresenta os resultados em modo texto. Além disso, permite que os dados de monitoramento sejam apresentados por gráficos e relatórios, com a utilização de uma ferramenta auxiliar chamada Flow-Scan [36].

2.2.5 NfSen/Nfdump

O NfSen/Nfdump [30] é uma ferramenta de código aberto capaz de realizar a coleta, armazenamento, processamento e apresentação dos dados de monitoramento. Os dados utilizados pelo NfSen são fluxos IP.

Esta ferramenta é utilizada geralmente para responder dois tipos de consultas: gerenciais e investigativas. Consultas gerenciais envolvem dados históricos, também conhecidos como dados *frios*, relativos a semanas, meses e anos. Elas fornecem informações sobre o comportamento da rede monitorada, atendendo questões como: qual a evolução da utilização do *link* de dados? Qual protocolo vem crescendo em utilização nas últimas semanas? Qual o tipo de tráfego mais trocado entre Sistemas Autônomos (do inglês, *Autonomous System - AS*). Este tipo de consulta auxilia na compreensão e planejamento dos investimentos destinados à infraestrutura de rede.

As consultas investigativas estão relacionadas a incidentes de segurança. Para estes casos, as consultas executadas visam investigar e encontrar os causadores de situações anômalas. Estas consultas geralmente são realizadas sobre dados recentes, ditos dados *quentes*. Elas também demandam menor tempo de resposta para o processamento das consultas e envolvem períodos curtos que duram entre 30 minutos e 12 horas. As consultas mais comuns para este objetivo respondem questões como: qual rede ou IP com maior tráfego? Qual porta e protocolo mais utilizados? Qual o motivo de haver muito tráfego de várias origens para um único destino?

Tanto consultas investigativas bem como consultas gerenciais usualmente utilizam o recurso de filtragem dos dados. Para o NfSen/Nfdump, o filtro é um parâmetro utilizado na consulta que pode ser aplicado sobre algumas das características dos fluxos IP, tais como: endereços IP de rede ou dispositivo, portas de comunicação, protocolos, tipos de serviço e quantidade de pacotes.

Embora seja amplamente conhecido apenas como NfSen, esta ferramenta é composta de duas ferramentas que atuam em conjunto: o próprio NfSen e o Nfdump. As ferramentas NfSen e Nfdump e seu funcionamento são descritos a seguir.

NfSen: O NfSen é responsável pela interface amigável com o usuário. Seu objetivo é fornecer recursos de visualização sobre os dados de monitoramento. Para isto, o administrador pode interagir com a ferramenta por meio de consultas pré-determinadas utilizando o recurso *profile* obtendo os resultados com o auxílio de gráficos e relatórios. Além disso, o NfSen provê mecanismos para que o usuário defina sua própria consulta, atuando em conjunto com as consultas pré-existentes no sistema.

A interface gráfica utilizada pelo administrador de rede para criar as consultas sobre os dados de monitoramento é apresentada na Figura 2.3.

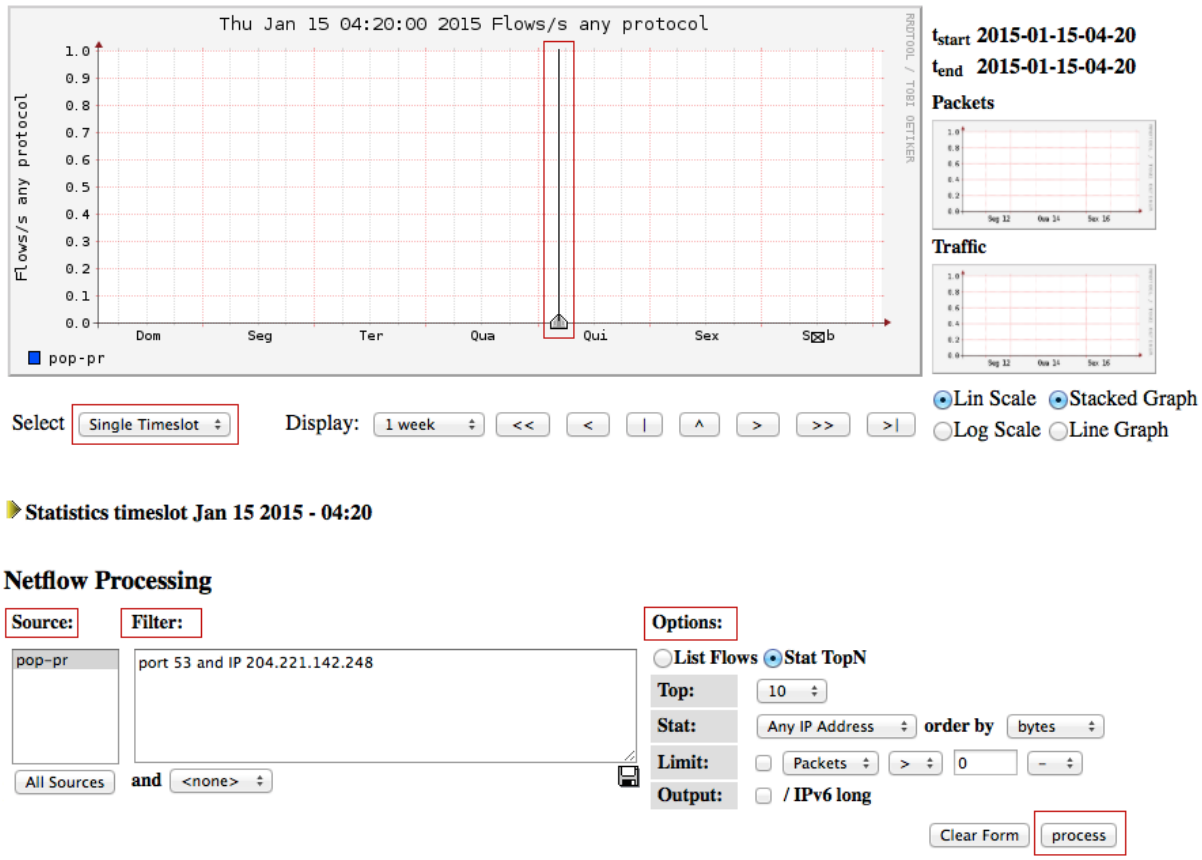

Figura 2.3: Recorte da interface de consulta da ferramenta NfSen.

Para elaborar a consulta, o administrador deve informar os valores desejados para os parâmetros de cada elemento que irá compor a consulta. Os elementos que compõem a consulta são: *origem, período, filtro* e *opções*. Inicialmente ele escolhe o período utilizando o botão *Single Timeslot/Time Window*. Em seguida, as origens são selecionadas por meio de um botão do tipo *multiple select* chamado *Source*. Após isso, o administrador pode

informar os filtros, se necessário, no campo *Filter*. Para finalizar, as opções de saída são selecionadas nos botões do grupo *Options*. Alguns dos parâmetros e valores aceitos em cada elemento da consulta é descrito a seguir juntamente com a ferramenta Nfdump.

Nfdump: O Nfdump é a ferramenta base para o funcionamento do NfSen. Ela é responsável pela manipulação dos fluxos IP. Isto é, ela realiza a coleta, trata o armazenamento e responde as consultas requisitadas pelo NfSen. Para a tarefa de coleta e armazenamento dos dados, o Nfdump agrupa as informações sobre o tráfego de rede de acordo com o tempo em que foram recebidas. Por exemplo, os dados contendo os fluxos IP recebidos no dia 15/02/2015 entre às 11:00 e 11:05 da manhã, são agrupados e armazenados em um arquivo chamado `nfcapd.201502151105`. Devido à maneira pela qual o Nfdump armazena os dados de monitoramento, pode-se atribuir aos arquivos a mesma classificação indicada aos dados de monitoramento. Isto é, pode-se classificar os arquivos como quentes ou frios, de acordo com o momento em que são gerados.

Para o processamento de consultas, o Nfdump pode atuar respondendo as requisições oriundas do NfSen ou com uma interface em modo texto onde o administrador de rede cria a consulta sem o auxílio do NfSen.

Toda consulta gerada a partir do NfSen é convertida para um comando. Este comando é enviado para o Nfdump para que ele o processe e retorne o resultado para o NfSen. A consulta convertida para o formato do Nfdump é formada a partir dos quatro elementos disponíveis no NfSen, citados anteriormente, que são:

[origem] [período] [filtro] [opções]

Cada elemento é composto de um parâmetro associado a um valor, que podem ser omitidos dependendo da consulta por possuírem valores padrão. O parâmetro indica um comportamento que será assumido em função do valor vinculado a este parâmetro. Para criar as consultas desejadas, o administrador de rede pode selecionar os parâmetros de cada elemento na tela do NfSen, ou se preferir, criar diretamente a linha de comando aceita pelo Nfdump. Neste caso, para cada elemento o usuário deve fornecer um ou mais parâmetros associados aos seus respectivos valores. Uma breve descrição de cada elemento é apresentada a seguir:

Origem: Determina quais são os exportadores (roteadores) envolvidos na consulta. Isto é, este elemento permite indicar sobre quais redes monitoradas deseja-se obter informações.

Período: Determina o período de tempo sobre o qual a consulta será aplicada. Por exemplo, é possível obter todos os fluxos IP compreendidos no período do dia 04/05/2015 a partir das 10:00 horas até 06/05/2015 às 14:00 horas. Neste caso, considera-se para a execução desta consulta uma sequência de arquivos iniciando com o arquivo `nfcapd.201505041000` e terminando com o arquivo `nfcapd.201505061400`.

Filtro: Para todos os arquivos envolvidos na consulta, este elemento indica que somente os fluxos IP que satisfazem os filtros da consulta devem ser retornados. Por exemplo, é possível criar uma consulta sobre os fluxos IP do arquivo `nfcapd.201503151105` que retorne apenas os que possuem informações de tráfego na porta 80 (HTTP). Outros valores aceitos neste elemento e usualmente utilizados são: rede, interface, endereço IP e protocolo de comunicação.

Opções: Para cada consulta, as opções neste elemento indicam o comportamento do resultado da consulta que será apresentado para o usuário. Os valores aceitos podem determinar por exemplo, a quantidade (com o parâmetro '-n') e ordenação (com o parâmetro '-s') dos fluxos IP que serão exibidos na tela. Outro valor possível para este elemento é com o parâmetro '-w nome_do_arquivo' que indica ao Nfdump que a saída será armazenado em um arquivo com o mesmo formato do arquivo de entrada.

A Figura 2.4 apresenta, já convertida para o formato aceito pelo Nfdump, a consulta exemplo da Figura 2.3 indicando os elementos e seus respectivos parâmetros e valores.

Consulta	-M /var/nfsen/profiles-data/live/pop-pr	-r nfcapd.201501150420	'port 53 and IP 204.221.142.248'	-n 10 -s ip/bytes
Elementos	[origem]	[período]	[filtro]	[opções]

Figura 2.4: Consulta exemplo do NfSen/Nfdump.

O parâmetro '-M' indica uma ou mais origens dos fluxos IP sobre os quais se deseja efetuar a consulta. Neste caso espera-se obter informações sobre o tráfego de rede da origem chamada **pop-pr**. O parâmetro '-r' do elemento período indica que a consulta será realizada apenas sobre os fluxos IP armazenados em um único arquivo. No elemento

filtro tem-se duas expressões: 'port 53' e 'IP 204.221.142.248'. Elas indicam que o resultado deverá conter apenas os fluxos IP que possuem como destino ou origem o IP 204.221.142.248 e tráfego na porta 53 (protocolo DNS). Neste elemento são aceitas uma ou mais expressões combinadas com os operadores *and, or e not*. Para o elemento opções, são fornecidos o parâmetro '-n 10' indicando a quantidade limite de até 10 resultados a serem exibidos na tela, e o parâmetro '-s ip/bytes', que diz para a ferramenta ordenar a apresentação considerando os endereços IP que mais possuem *bytes* de tráfego. A Figura 2.5 apresenta o resultado da execução da consulta exemplo[3].

```
Top 10 IP Addr ordered by bytes:
Date first seen          Duration Proto          IP Addr    Flows(%)      Packets(%)       Bytes(%)         pps      bps   bpp
2015-01-15 04:20:12.313   298.983 any     204.221.142.248      73(100.0)      120(100.0)      8611(100.0)        0      230    71
2015-01-15 04:23:29.024   102.272 any       132.128.87.82       41(56.2)       41(34.2)       3377(39.2)        0      264    82
2015-01-15 04:20:20.754   256.546 any       143.34.86.229        8(11.0)       24(20.0)       1536(17.8)        0       47    64
2015-01-15 04:20:32.617   256.329 any     130.210.140.126        4( 5.5)       24(20.0)       1536(17.8)        0       47    64
2015-01-15 04:21:44.833   126.140 any     132.231.116.228       11(15.1)       20(16.7)       1326(15.4)        0       84    66
2015-01-15 04:21:37.547   123.091 any     130.160.24.246         4( 5.5)        6( 5.0)        456( 5.3)        0       29    76
2015-01-15 04:20:12.313   132.349 any     204.210.13.198         5( 6.8)        5( 4.2)        380( 4.4)        0       22    76

IP addresses anonymised
Summary: total flows: 73, total bytes: 8611, total packets: 120, avg bps: 230, avg pps: 0, avg bpp: 71
Time window: 2015-01-15 04:20:12 - 2015-01-15 04:25:11
Total flows processed: 1011905, Blocks skipped: 0, Bytes read: 75019624
Sys: 0.174s flows/second: 5798483.8  Wall: 0.174s flows/second: 5811771.7
```

Figura 2.5: Resultado da consulta exemplo do NfSen/Nfdump.

A lista completa com os valores possíveis para cada um dos elementos é apresentada no Apêndice A.

O NfSen/Nfdump é muito utilizado em diversas instituições por ser gratuita e possuir ampla documentação. Esta ferramenta no entanto, possui limitações relacionadas à escalabilidade porque atua de modo centralizado nas etapas de coleta, processamento e armazenamento dos fluxos IP.

2.3 Análise das Ferramentas de Monitoramento de Redes

A Tabela 2.1 apresenta de modo resumido uma comparação entre as ferramentas de monitoramento e os conceitos apresentados anteriormente.

Considerando a descrição das ferramentas de monitoramento de redes, podem-se destacar importantes diferenças entre elas. A começar, observa-se uma relação entre o dado monitorado e o mecanismo utilizado para a coleta. As ferramentas que utilizam o meca-

[3]Os endereços IPs foram anonimizados para preservar a confidencialidade dos usuários.

	Tcpdump	**Wireshark**	**Ntop**	**Flow-tools**	**NfSen / Nfdump**
Coleta	Captura	Captura	Recebimento	Recebimento	Recebimento
Dados de Monitoramento	Pacotes IP	Pacotes IP	Fluxos IP	Fluxos IP	Fluxos IP
Armazenamento e Arquitetura	Centralizado	Centralizado	Centralizado	Centralizado	Centralizado
Visualização	Modo Texto	Modo Gráfico	Modo Gráfico	Modo Texto	Modo Texto e Modo Gráfico
Consultas	Dinâmica	Dinâmica	Pré-definida	Dinâmica	Dinâmica e Pré-definida

Tabela 2.1: Tabela comparativa entre as ferramentas de monitoramento.

nismo de captura, trabalham com dados de monitoramento considerados brutos (pacotes IP). Em contrapartida, as ferramentas configuradas para tratar os fluxos IP, efetuam a coleta por meio de recebimento. Nota-se portanto que o Tcpdump e Wireshark demandam mais trabalho por parte do administrador para efetuar uma análise da rede monitorada. As demais ferramentas são acessíveis também a usuários menos experientes.

Além disso, percebe-se que o acesso aos dados monitorados em algumas ferramentas é feito em modo texto. Isto implica em uma visualização pouco amigável e a análise da rede compete a um usuário com experiência. Isto ocorre porque os dados resultantes de uma consulta ainda não são tratados e intuitivos. Sob este aspecto, Ntop e NfSen/Nfdump oferecem uma melhor experiência de utilização ao apresentar os dados em modo gráfico.

Outro ponto importante é referente às consultas. A existência de um conjunto pré-determinado de filtros permite a utilização da ferramenta por usuários com diferentes perfis. Entretanto, este modo restringe o acesso a dados que possam ser úteis e são obtidos somente com a montagem de uma consulta dinâmica. Neste sentido, o NfSen/Nfdump permite que sejam extraídas informações adequadas para cada consulta. Isso porque esta ferramenta oferece em uma mesma interface a utilização de consultas pré-determinadas em conjunto com filtros dinâmicos.

Em relação à arquitetura, nota-se que todas as ferramentas analisadas possuem a mesma forma de armazenamento, feito de modo centralizado. Considerando a necessidade

atual da retenção dos dados de monitoramento [11, 31], constata-se que nenhuma das ferramentas apresentadas é realmente escalável. Isto significa que mesmo sendo necessária, a retenção dos dados de monitoramento fica restrita à capacidade de armazenamento do disco local.

Adicionalmente, a arquitetura centralizada possui algumas deficiências inerentes a este modelo. Entre alguns dos problemas, pode-se destacar o ponto único de falha. Sob este aspecto, um armazenamento centralizado não dispõe de mecanismos de redundância, ou seja, um problema fatal no disco ou no servidor implica na perda dos dados. Além disso, tem-se o problema da sobrecarga de processamento. Um número excessivo de acessos ao servidor pode gerar uma falha. Isto ocorre porque no modelo centralizado não existe divisão de tarefas.

Considerando esta análise, no próximo capítulo serão discutidos alguns trabalhos relacionados à escalabilidade para a atividade de monitoramento de rede. A escalabilidade proposta nos trabalhos é fornecida visando um dos aspectos do monitoramento, isto é, para coleta, armazenamento, processamento e apresentação dos dados. Para isto, os trabalhos apresentam diversas abordagens para os problemas da arquitetura centralizada, dentre as quais, destaca-se a abordagem distribuída.

CAPÍTULO 3

TRABALHOS RELACIONADOS

A escalabilidade de monitoramento é descrita como a capacidade de contornar os problemas relacionados ao modelo centralizado. Isto significa que uma solução escalável deve oferecer condições para que as tarefas de coleta, armazenamento e processamento possam ser realizadas independentemente da quantidade de dados manipulados.

Neste capítulo são apresentados alguns trabalhos que abordam o problema da escalabilidade em ambientes de monitoramento de rede. Os trabalhos são discutidos de acordo com a abordagem utilizada. Os primeiros estão relacionados com a escalabilidade na etapa de coleta e captura dos dados de monitoramento. Na sequência, são descritos os trabalhos que atuam na fase de armazenamento dos fluxos IP. Por fim, são apresentadas as soluções que focam no processamento, recuperação e apresentação dos dados.

3.1 Coleta

Para a tarefa de coleta dos dados de monitoramento, existem soluções que buscam executar esta etapa em ambientes com grandes quantidades de dados. Para isso, apresentam soluções que utilizam a abordagem centralizada e distribuída. Em ambos os casos, o objetivo é dar suporte às ferramentas, a fim de que elas possam efetuar a captura com o mínimo ou nenhuma perda dos dados de monitoramento.

3.1.1 n2disk

Deri et al. (2013) [9] apresentam uma aplicação chamada **n2disk** que é capaz de capturar dados de monitoramento em redes de 10Gbps. Atuando no papel de coletor, a ferramenta tem como objetivo coletar pacotes IP em redes onde trafegam grandes volumes de dados. O principal desafio desta ferramenta é desempenhar a tarefa utilizando dispositivos comuns e de baixo custo. O seu funcionamento consiste na premissa de que nem todos os

dados são úteis e devem ser coletados. Por conta disso, o utilizador cria filtros para indicar quais tipos de dados devem ser capturados e armazenados pela ferramenta. Estes filtros são aplicados sobre informações contidas nos cabeçalhos dos pacotes IP, dentre os quais encontram-se: IP origem, IP destino, protocolo e porta de comunicação. Quando o exportador envia os dados de monitoramento, o n2disk recebe-os, armazena temporariamente em um *buffer* de memória e em seguida aplica os filtros definidos pelo utilizador. Após isso, os dados de monitoramento que satisfazem os filtros são indexados e armazenados permanentemente em disco.

Os benefícios desta solução estão associados à simplicidade da arquitetura utilizada para efetuar a coleta, uma vez que ela procura manter o modelo clássico de um coletor para vários exportadores. Adicionalmente, o trabalho propõe uma solução capaz de realizar a tarefa de coleta com eficiência utilizando computadores de baixo custo e poder computacional. Este trabalho porém, tem como objetivo tratar grandes volumes de dados apenas na etapa de coleta. Com isso, as tarefas de processamento e armazenamento não são atendidas pela solução.

3.1.2 DiCAP

O trabalho de Morariu e Stiller (2008) [28] apresenta uma abordagem direcionada à captura distribuída dos dados de monitoramento. Neste trabalho, cria-se um agrupamento de coletores chamado de *cluster* de captura. No *cluster*, cada coletor possui uma identificação única e todos estão interligados entre si por meio de uma rede *Peer-to-Peer* (P2P). Além do *cluster*, existem dois elementos que compõe a arquitetura: o nó coordenador e o servidor que possui uma ou mais ferramentas de monitoramento. O nó coordenador tem a função de controlar todos os demais para que eles trabalhem em conjunto sem sobreposição de atividades, ou seja, dois coletores não capturam os mesmos dados. Isto é possível porque cada nó recebe um conjunto de regras do coordenador. Estas regras indicam para cada coletor o que deve ser capturado evitando que haja duplicidade.

O processo de captura no *cluster* é o diferencial a fim de obter escalabilidade de coleta. Para isso, os autores criaram uma aplicação que atua diretamente sobre a interface de

rede. Esta aplicação executa uma filtragem no cabeçalho dos pacotes IP baseada nas regras recebidas pelo nó coordenador. A etapa de coleta ocorre da seguinte maneira: os exportadores enviam uma cópia de todo o tráfego para todos os coletores. Este tráfego chega então à interface de rede de cada nó. Os coletores efetuam a leitura dos cabeçalhos de cada pacote IP a fim de determinar quais coincidem com as regras. Os pacotes aceitos são enviados a uma segunda aplicação que faz o armazenamento do pacote completo, ou seja, conteúdo e cabeçalho. Após isso, os dados são disponibilizados para o servidor que possui a ferramenta de monitoramento a fim de que esta possa utilizá-los.

O princípio de operação desta solução envolve a utilização de uma DHT (*Distributed Hash Table*). Uma DHT é uma forma de distribuir informações em uma rede na qual os dados são identificados por um par: (*chave*, *valor*). A localização de um dado é determinada pelo resultado de uma função *hash* aplicada sobre a *chave* [33]. Com essa técnica, o nó coordenador determina quais dados devem ser capturados por cada coletor baseados na sua identificação no *cluster*.

Embora a solução proposta neste trabalho forneça escalabilidade para a tarefa de coleta com a utilização de um *cluster* de captura e DHT, ela não trata das etapas de processamento e armazenamento.

3.2 Armazenamento

Alguns trabalhos propõem alternativas para tratar o armazenamento de grandes volumes de dados de monitoramento. Ainda que estas soluções tenham afinidade em relação ao objetivo final, cada uma possui sua particularidade, seja na manipulação dos dados ou na forma como fazem o armazenamento. Para qualquer situação, uma solução considerada escalável deve permitir o armazenamento dos dados levando em consideração um aumento no volume que não é previamente conhecido.

3.2.1 TIFA

Li et al. (2011) [24] propõem o sistema TIFA cujo objetivo é armazenar e recuperar grandes volumes de dados de monitoramento em tempo real. Isto é, os dados devem estar disponíveis para as ferramentas de monitoramento à medida que são coletados. Para isso, a solução divide a tarefa em duas etapas. A primeira, consiste na utilização de uma aplicação chamada *Time Machine* que é executada no coletor para efetuar o recebimento dos dados provenientes dos exportadores. No passo seguinte, uma segunda aplicação, chamada *FastBit*, indexa os dados de monitoramento recebidos. Após a indexação, a solução agrupa as informações em um único arquivo para em seguida armazená-los em disco. Com o intuito de aumentar a capacidade de armazenamento, esta solução é capaz de gerenciar a utilização de vários discos simultaneamente. Uma vez armazenados, os dados de monitoramento são recuperados com a extensão "*.pcap*". Desta forma, podem ser utilizados por ferramentas de monitoramento de rede que manipulam este formato de arquivo, como por exemplo o **Tcpdump** e **Ntop**.

Observa-se que neste modelo a arquitetura tradicional de monitoramento é preservada, ou seja, há um coletor para vários exportadores. Ainda que a solução aumente a capacidade de armazenamento ao manipular diversos discos, ela foi planejada para atuar em um único coletor. Desta forma, a escalabilidade de coleta e armazenamento está limitada à capacidade da máquina na qual a aplicação está sendo executada.

3.2.2 DIPStorage

No trabalho de Morariu et al. (2008) [26], os autores propõem uma arquitetura distribuída para o armazenamento de fluxos IP. Esta proposta é semelhante ao DiCAP [28] no quesito arquitetura, pois é baseada em uma rede P2P e funciona com a utilização de uma DHT. A solução é composta por um conjunto de nós interligados atuando no papel de coletor. Cada coletor possui uma aplicação responsável por capturar fluxos IP e um conjunto de regras para determinar como eles serão distribuídos. As regras tem o objetivo agrupar as informações de monitoramento de acordo com seus conteúdos. Elas atuam da seguinte

maneira: a cada fluxo IP que chega a um servidor, este aplica uma função *hash* sobre um dos campos do cabeçalho de acordo com as regras que possui, por exemplo: IP origem, IP destino, porta origem, porta destino. O resultado da função *hash* determina qual o servidor responsável por armazenar as informações do fluxo IP.

A manipulação dos dados é feita com o objetivo de otimizar o desempenho das consultas. Isto é feito porque o armazenamento em um ambiente distribuído com DHT a aplicação não controla a localidade dos dados armazenados. Neste sentido, os autores propuseram uma organização lógica dos servidores para armazenar os fluxos IP de acordo com as consultas mais utilizadas. Isto é, eles consideraram que as consultas mais executadas envolvem endereços IP de origem e destino e portas de comunicação. Assim, são formados agrupamentos virtuais de servidores para o armazenamento dos dados de acordo com tais parâmetros. Por exemplo, um conjunto de servidores $C1$ {S1, S2, S3 e S4} armazenam os fluxos de acordo com o endereço IP de origem. O conjunto $C2$ {S1, S5, S3 e S6} armazenam os fluxos agrupados pela porta de comunicação de destino, e assim por diante.

O agrupamento virtual dos servidores em conjuntos permite que as consultas mais comuns sejam executadas com menor tempo de resposta. Em contrapartida, as consultas menos comuns são atendidas com maior tempo de resposta. Por conta do padrão existente nas consultas aplicadas sobre os dados de monitoramento, os autores entendem que o ganho no desempenho das consultas mais comuns compensam o atraso em outras não tão utilizadas.

Este trabalho obtém escalabilidade de armazenamento com a utilização de uma DHT. A solução proposta pelos autores permite que nodos sejam adicionados ao ambiente a qualquer tempo a fim de impedir que o espaço de armazenamento seja esgotado. Contudo, a solução apresentada não contempla a tarefa de coleta.

Adicionalmente, os critérios utilizados para a organização dos fluxos armazenados não consideram o tempo em que a informação foi gerada. Com isso, os fluxos antigos e recentes de um determinado endereço IP, por exemplo 10.10.10.1, são armazenados no mesmo conjunto de servidores. Assim, uma consulta sobre este endereço IP que deseja obter

informações sobre um dia anterior, deve aguardar o processamento de todos os fluxos que possuem este endereço IP, indiferentemente do período em que o fluxo foi gerado. Desta forma, a consulta pode ser penalizada com um acréscimo no tempo de resposta.

3.2.3 MicroCloud-based

Deri e Fusco (2013) [10] propõem uma arquitetura de armazenamento para os dados de monitoramento baseada em um conjunto de servidores interligados entre si denominado "micro nuvem". O principal objetivo desta estrutura é tornar os dados acessíveis a várias ferramentas de monitoramento simultaneamente e em tempo real. O conceito chave para tornar isto possível é permitir que os exportadores acessem o ambiente para escrita ao mesmo tempo em que as ferramentas o acessam para leitura. Para isto os autores dividiram as funções dos exportadores, coletores e ferramentas de monitoramento em três elementos distintos da arquitetura. Os coletores são os servidores que compõem a micro nuvem propriamente dita. Para implementá-la eles utilizaram o Redis[1]. Esta aplicação permite criar um ambiente de armazenamento distribuído baseado em uma DHT.

As ferramentas de monitoramento pode estar executando em qualquer dispositivo que possua acesso à micro nuvem. Por utilizar como forma de armazenamento um par (chave, valor), cada ferramenta necessita saber apenas por qual chave deve buscar no ambiente. Este por sua vez se encarrega de encontrá-la e retornar o conteúdo para a ferramenta que o solicitou.

Para atingir o objetivo principal, os autores atribuíram aos exportadores um papel fundamental desta arquitetura. Na maioria das arquiteturas de monitoramento, os roteadores são os elementos exportadores de fluxos IP. Porém, nesta solução, os exportadores são computadores convencionais executando uma aplicação desenvolvida pelos próprios autores chamada nProbe [7]. O nProbe é um programa criado para gerar fluxos IP a partir do tráfego bruto de rede. Ele é uma aplicação configurável e pode gerar os fluxos de todo o tráfego ou de parte dele. O nProbe funciona em três estágios: inicialmente recebe uma cópia de todo o tráfego que passa pela rede; a partir disso, ele analisa os dados baseados

[1] http://redis.io

em critérios definidos pelo utilizador a fim de determinar de que informações deve gerar os fluxos IP; em seguida, gera os fluxos e envia-os para os coletores.

O diferencial desta solução está na seleção de quais dados brutos de rede serão gerados os fluxos IP. Por ser customizável, o utilizador define no nProbe quais são os fluxos IP de interesse baseados nas ferramentas que serão utilizadas. Assim, os dados de monitoramento são criados de modo personalizado, isto é, somente os dados de interesse são enviados e armazenados na micro nuvem. Desta forma, otimiza-se os recursos computacionais do ambiente tanto para leitura como escrita dos dados de monitoramento.

Com esta arquitetura os autores obtiveram uma solução escalável para o armazenamento. Isto porque na implementação da micro nuvem é possível adicionar nodos ao ambiente a qualquer momento. A desvantagem desta solução é a ausência do controle de localidade, por parte da aplicação, sobre os dados armazenados. Isto ocorre porque este trabalho utiliza uma rede com DHT. Esta técnica não fornece mecanismos para que a aplicação obtenha o controle de localidade dos dados armazenados, podendo, desta forma, impactar negativamente no desempenho das consultas.

3.2.4 Internet Traffic Analysis with MapReduce

Lee et al. (2010) [22] apresentam uma abordagem de processamento e armazenamento para grandes volumes de dados de monitoramento com o Hadoop [17]. O objetivo principal é permitir que uma ferramenta de monitoramento de rede consiga processar e armazenar *TeraBytes* ou *PetaBytes* de fluxos IP. Para isso, é proposto um conjunto de servidores que atuam como coletores. Eles são coordenados por um nó conhecido como *master node*. O funcionamento da arquitetura começa pela coleta dos dados. Os exportadores enviam os fluxos IP para qualquer coletor ativo. Cada coletor é parte de um sistema de arquivos distribuído, chamado *Hadoop Distributed File System* (HDFS). À medida que os dados são recebidos pelos coletores, eles os processam e armazenam seguindo o modelo de *MapReduce*[2]. Este processamento tem como objetivo tornar os dados aptos a serem armazenados no HDFS. Isto é necessário porque os dados de monitoramento em

[2]http://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/

seu formato original são incompatíveis com o HDFS.

A principal contribuição desta solução é a aplicação do modelo *MapReduce* para o armazenamento e processamento. Contudo, a solução não trata da escalabilidade para a atividade de coleta dos dados de monitoramento de rede.

3.3 Processamento, Recuperação e Apresentação

A escalabilidade nas etapas de processamento, recuperação e apresentação requer que os dados sejam manipulados de modo que a ferramenta de monitoramento de rede consiga atender às requisições da mesma forma como o fazia em ambientes com pequenos volumes de dados. Para isso, os trabalhos a seguir descrevem soluções que visam aumentar a capacidade das ferramentas para estas etapas.

3.3.1 SCRIPT

Morariu et al. (2010) [27] apresentam um *framework* para processar grandes volumes de fluxos IP em tempo real. O objetivo principal consiste em permitir que ferramentas distintas de monitoramento tenham acesso, de forma transparente, aos dados que são capturados e processados de modo distribuído. Para isso, é proposta uma arquitetura baseada em DHT com uma rede P2P. Nesta rede, todos os nós são coletores e podem comunicar-se uns com os outros por meio de uma aplicação proposta pelos autores e que compõe o *framework*. Cada coletor é responsável por capturar uma porção do tráfego de rede. Além disso, cada servidor é utilizado para manter em operação uma ou mais ferramentas de monitoramento.

Os fluxos coletados são submetidos a um conjunto de regras existentes em cada nó. Estas regras são fornecidas por cada ferramenta de monitoramento que está presente no ambiente. O objetivo das regras é otimizar tanto o armazenamento quanto a recuperação dos dados de monitoramento em virtude do tipo de dado que a aplicação utiliza. Para exemplificar, considera-se uma ferramenta que é encarregada de monitorar o tráfego do protocolo *Domain Name Service* (DNS). Esta ferramenta cria uma regra indicando a

todos os coletores que um fluxo que contém este protocolo, deve ser armazenado o mais próximo possível do servidor onde a ferramenta está sendo executada. Para que as regras sejam disponibilizadas adequadamente a cada servidor, está presente na arquitetura um servidor com a função especial de coordenador. Ele é o responsável por enviar as regras de operação e demais informações de controle para todos os nós da rede, como por exemplo, mensagens com informações sobre a atividade de cada nó.

Este trabalho propõe um aumento no poder de processamento dos fluxos IP, porém, mantém a figura de um servidor central com funções importantes para o sistema. Além disso, a solução utiliza os recursos da DHT com uma rede P2P, logo, as aplicações possuem o controle da localidade de forma aproximada dos dados armazenados.

3.3.2 Toward Scalable Traffic with Hadoop

Lee e Lee (2013) [21] apresentam uma proposta de arquitetura que trata da recuperação e apresentação dos dados de monitoramento no ambiente Hadoop. Esta solução tem como diferencial a capacidade de armazenar e recuperar tanto pacotes IP quanto fluxos IP. Para isso, os autores apresentam uma arquitetura que recupera os dados armazenados no HDFS de acordo com a ferramenta de monitoramento que os solicita. Para esta solução, cada nó do ambiente é parte ativa do sistema de arquivos distribuído (HDFS). Eles são acessados por um servidor central que envia as consultas para serem processadas. Neste momento, o conjunto de servidores inicia o processo de recuperação dos dados em função do tipo de ferramenta que os solicitou. Por exemplo, se a ferramenta que deseja o dado é o NfSen, os dados são recuperados no formato de um fluxo IP. Além do armazenamento e recuperação, é proposta neste trabalho uma interface de consultas que auxilia no processo de monitoramento.

Esta solução fornece uma maior capacidade de armazenamento. Para isso, faz uso de um sistema de arquivos distribuído para o armazenamento. Esta abordagem é relevante sob o aspecto de otimização de consultas, uma vez que pode-se controlar a localidade dos dados. Entretanto, a solução não trata da atividade de coleta dos dados de monitoramento.

3.4 Sumário

Como pode-se observar nos trabalhos apresentados, há forte interesse em prover escalabilidade para as ferramentas de monitoramento. Alguns procuram otimizar e garantir o desempenho na etapa de coleta. Outros focam na forma como será feito o armazenamento a fim de fornecer um maior espaço para armazenar os dados. Ainda, existem trabalhos que buscam aproveitar os recursos para desempenhar mais de uma tarefa no mesmo ambiente.

Neste sentido, nota-se que grande parte dos esforços estão direcionados para arquiteturas distribuídas. Isto porque os ambientes distribuídos podem ser expandidos a fim de evitar o esgotamento de recursos. Porém, os trabalhos criam estruturas que dependem de elementos centrais para responder por tarefas importantes do sistema. Sejam elas utilizadas nas etapas de coleta, armazenamento ou processamento e apresentação. Para criar o ambiente distribuído, observa-se a utilização de redes P2P e sistemas de arquivos distribuídos. As redes P2P com a utilização de DHTs são técnicas de fácil implementação e que operam de modo muito eficiente. Elas possuem como desvantagem o fato de não permitir que as aplicações controlem a localidade exata dos dados armazenados. Este é um recurso muito utilizado para otimizações de consultas [35]. Sob este aspecto, os sistemas de arquivos distribuídos podem oferecer esta funcionalidade como diferencial em relação às DHTs. Neste sentido, no próximo capítulo são apresentados conceitos e funcionalidades a respeito dos sistemas de arquivos distribuídos mais utilizados atualmente.

CAPÍTULO 4

SISTEMAS DE ARQUIVOS DISTRIBUÍDOS

Os sistemas de arquivos distribuídos são sistemas muito utilizados para armazenamento de grandes volumes de dados. Estes sistemas permitem que diversas aplicações, incluindo as ferramentas de monitoramento de rede, possam obter escalabilidade de armazenamento. Neste sentido, este capítulo apresenta as principais definições e conceitos sobre os sistemas de arquivos distribuídos. Na sequência, descreve-se as características dos sistemas utilizados atualmente. Ao final, é feita uma análise comparativa dos sistemas apresentados.

4.1 Definições

Segundo Levy e Silberschatz [23], os sistemas de arquivos distribuídos (do inglês, *Distributed File System - DFS*), são sistemas que permitem o armazenamento permanente de arquivos de modo distribuído. Isto é, o armazenamento de arquivos em um DFS é realizado por um ou mais dispositivos conectados entre si por meio de uma rede de comunicação. Esses sistemas devem dar suporte a operações de criação, escrita, leitura e exclusão do mesmo modo que os sistemas de arquivos tradicionais ou centralizados. No contexto dos DFS, podem ser identificados três componentes principais:

Serviço: Aplicação (*software*) executado em um ou mais nós do ambiente distribuído;

Servidor: Nó (nodo) do ambiente distribuído que executa um serviço (uma aplicação) de armazenamento; e

Cliente: Usuário que interage com o sistema a fim de obter recursos providos pelo serviço.

Por definição, um DFS deve ser capaz de oferecer alguns recursos presentes nos sistemas de arquivos tradicionais (ou centralizados), tais como: **transparência**, **tolerância a falhas** e **escalabilidade** [23].

A transparência é composta por dois conceitos: transparência de localidade e localização independente. Na transparência de localidade, um usuário não sabe em qual nodo da rede o arquivo está armazenado. Já na localização independente, um nome global único é dado para o arquivo em todo o sistema. Essa propriedade está relacionada ao processo de migração dos dados entre servidores. Desta forma o nome do arquivo fica desassociado do servidor, permitindo livre trânsito dos dados entre os nodos do sistema sem afetar a aplicação.

Com relação à tolerância a falhas, o DFS deve continuar fornecendo o serviço para o cliente mesmo em casos de falhas. Este conceito está associado à disponibilidade de um sistema de arquivos distribuído. As falhas podem ser de diversos tipos, tais como: falha na rede, falha em um servidor e parada no serviço. Ainda em relação às falhas, o DFS deve considerar que elas podem ser permanentes ou temporárias. A falha temporária ocorre quando um elemento falho se recupera e volta a participar do sistema.

Em termos de escalabilidade, considera-se a capacidade do sistema de dar suporte a um crescimento em sua carga de trabalho. Esse aumento de trabalho deve ser absorvido sem causar prejuízos ao funcionamento do DFS.

4.2 Conceitos

Os DFS modernos são descritos por conceitos que visam disponibilizar os recursos citados na seção 4.1, entre os principais conceitos tem-se: nomeação, arquitetura de metadados, detecção de falhas, replicação e política de localização, controle de localidade, sincronização e consistência, balanceamento de carga e API. Estes conceitos são descritos a seguir.

4.2.1 Nomeação

Nomeação é o mapeamento entre o nome do arquivo conhecido para o cliente e seu identificador interno no sistema. Em sistemas tradicionais, uma aplicação conhece um arquivo pelo seu nome textual. Nos sistemas de arquivos distribuídos, este mapeamento passa a ser em qual nó da rede o dado está armazenado e como é encontrado. A maneira mais utili-

zada pelos DFS para prover a nomeação é com o emprego de um nome global único. Deste modo, o sistema garante o fundamento de transparência, isto é, o processo de migração de arquivos entre servidores é abstraído da aplicação [23]. Outra forma de prover nomeação é com a combinação: nome do servidor e nome do arquivo (ex. servidor1:arquivo1). Desta maneira, a propriedade de localização independente é quebrada. Embora o nome do arquivo continue sendo único, sua migração na rede afetaria a aplicação.

A maioria dos sistemas atuais fornece nomeação com a utilização de diferentes mecanismos, tais como: indexação, base de dados e algoritmos. A indexação consiste em uma estrutura interna com informações sobre os arquivos armazenados no DFS [23]. Essas informações, também conhecidas como **metadados**, são os atributos dos arquivos. Os atributos mais comuns são: nome do arquivo, proprietário, localização, tamanho, descrição e data de criação. Essa estrutura é armazenada como um registro interno do DFS [17, 18].

Seguindo na mesma linha, uma base de dados pode ser utilizada para armazenar as informações de metadados. O conceito sobre os atributos dos arquivos é equivalente, porém, seu armazenamento não é feito como registro interno do DFS, mas em uma base de dados convencional (ex. MySQL, PostgreSQL) [20].

Outra abordagem utilizada é por meio de algoritmos de nomeação [15, 42]. Nesta técnica, os atributos de um arquivo o acompanham durante sua existência no DFS. A principal diferença deste conceito para os demais, consiste em não depender de uma estrutura de metadados para o controle de nomeação e localização dos arquivos.

4.2.2 Arquitetura da Gerência de Metadados

A arquitetura de um sistema de arquivos distribuído é categorizada pela forma com a qual o DFS gerencia os metadados. Conforme descrito na seção 4.2.1, os metadados são informações sobre os atributos dos arquivos. Sob este aspecto, os servidores de metadados podem estar organizados de maneira centralizada ou distribuída [40].

Quando centralizada, a arquitetura é caracterizada por possuir um único servidor responsável por armazenar e coordenar as ações relacionadas aos metadados. Somado a

este, existe um conjunto de outros servidores que são responsáveis por armazenar os dados propriamente ditos.

Por outro lado, a arquitetura distribuída permite que todos os servidores do ambiente desempenhem o mesmo papel. Estes servidores podem ser utilizados para armazenar os metadados, os dados em si, ou ambos, sendo este último a abordagem mais utilizada.

A configuração de arquitetura escolhida por um DFS tem consequências sobre o conceito de transparência e tolerância a falhas. Um sistema que possui arquitetura centralizada utiliza um servidor central para gerenciar a localidade e nomeação dos arquivos armazenados. A principal vantagem desta abordagem é conhecer qual servidor é responsável por esta tarefa. Em contrapartida, essa arquitetura possui um ponto único de falha. Embora os dados não sejam perdidos, o sistema pode ficar indisponível temporariamente.

Para contornar este problema, foi desenvolvida a arquitetura distribuída. Nesta abordagem, todos os servidores da rede são capazes de determinar a localidade de um arquivo. Além disso, este modelo não depende de um servidor estratégico para manter-se em operação, garantindo desta forma, uma maior disponibilidade ao DFS. Contudo, a arquitetura distribuída possui um custo associado à distribuição, replicação e consistência do metadados.

4.2.3 Detecção de Falhas

Sob a perspectiva dos sistemas de arquivos distribuídos, a capacidade do DFS de identificar uma falha no sistema é denominada detecção de falhas. Um sistema precisa detectar uma falha para que ele possa tomar uma decisão sobre este evento. A detecção pode ser categorizada como sendo automática ou manual [6].

Uma das técnicas de detecção de falhas considerada automática é por meio de troca de mensagens. O DFS possui como regra de operação em cada um dos nós, um processo em que todos os participantes enviam mensagens para sinalizar que estão ativos. O não envio ou recebimento destas mensagens por algum nodo, permite ao DFS colocá-lo em inatividade temporária, e, após um determinado tempo que varia entre sistemas, excluí-lo do ambiente.

A detecção manual ocorre quando um serviço é solicitado a um determinado nodo, e este por sua vez não atende a requisição. A partir desse momento, o DFS inicia o processo de deixar o nodo inativo e posteriormente desativá-lo do sistema. A detecção de falhas em um DFS é uma das etapas para que este de suporte ao conceito de tolerância a falhas.

4.2.4 Replicação e Política de Localização

Uma forma de garantir que um DFS, mesmo com falhas, continue fornecendo arquivos íntegros para as aplicações, é utilizando o conceito de replicação. Em sistemas de arquivos distribuídos, a replicação consiste em manter várias cópias do mesmo arquivo em diferentes localidades. Para que isto seja possível, é preciso que haja uma política de localização [6]. A política de localização define o modo pelo qual as réplicas dos arquivos são distribuídas no sistema. Tanto a replicação quanto a localização das cópias estão presentes nos DFS e são categorizadas de duas maneiras: automática e manual.

Considera-se automática a replicação quando ela inicia o processo de criação e envio de cópias sem a intervenção do cliente. Além disso, por meio de uma política de localização, o sistema é capaz de determinar automaticamente para onde as cópias devem ser enviadas. Em contrapartida, na replicação manual é o cliente quem deve dar início ao processo. Neste caso ainda, o usuário deve indicar em quais nodos as cópias devem ser mantidas, implementando manualmente a política de localização.

Em ambos os casos, o sistema espera que seja fornecida uma quantidade de cópias dos arquivos. Se a replicação for manual, no momento em que o usuário dispara o processo, ele fornece a quantidade. Já na replicação automática, o sistema sabe antecipadamente quantas cópias são esperadas do arquivo.

Embora a replicação seja baseada em cópias de arquivos, na maioria dos casos, existe ainda a replicação de volumes [14]. Um volume é considerado uma unidade de armazenamento do sistema (ex. disco, partição). O mecanismo de replicação provido pelo DFS garante que ele implemente o conceito de tolerância a falhas.

4.2.5 Controle de Localidade

Em sistemas de arquivos distribuídos, a localidade de um arquivo armazenado influencia no desempenho das consultas [35]. Isto acontece porque quando um conjunto de dados de interesse (ex. pessoas, cidades) está armazenado em um mesmo servidor, toda a consulta pode ser processada localmente. Desta forma, somente o resultado da consulta esperado é enviado pela rede. Por conta disso, é importante que o DFS ofereça algum mecanismo para que este controle possa ser feito pelo utilizador do sistema e este controle deve ser dado de forma explícita. Isto significa que o cliente pode indicar com precisão em qual servidor determinados arquivos serão armazenados.

4.2.6 Sincronização e Consistência

A replicação de arquivos em um DFS está relacionada aos conceitos de sincronização e consistência. A sincronização determina o modo como o DFS mantém suas réplicas. Ela é classificada de três formas: síncrona, assíncrona e semi-assíncrona [6, 40].

Uma replicação do tipo síncrona impede que o arquivo seja acessado enquanto suas réplicas são atualizadas. Uma vez concluído o processo, o DFS retorna para a aplicação um estado de sucesso. Desta forma, garante que a última versão de um arquivo é entregue para o cliente quando solicitado. Todavia, o tempo de resposta neste modelo torna-se maior.

Ao contrário do modo síncrono, a replicação assíncrona permite que os dados sejam acessados enquanto são atualizados. Embora traga um tempo de resposta menor nas requisições, neste modelo os clientes podem eventualmente acessar alguma cópia do arquivo que não foi atualizada.

Definida como conceito intermediário entre os anteriores, a replicação semi-assíncrona bloqueia o acesso (leitura/escrita) sobre um arquivo até que algumas de suas cópias estejam atualizadas. Por exemplo, se um determinado arquivo possui 4 cópias, o sistema bloqueia o acesso a este arquivo até que 2 das 4 cópias tenham sido atualizadas. Com isso, diminui o tempo de resposta frente a replicação síncrona, pois permite o acesso aos

arquivos mesmo que alguns ainda estejam em processo de atualização. Adicionalmente, esse modelo diminui as chances de retornar arquivos inconsistentes para o cliente.

Associado ao modo de sincronização, o conceito de consistência define como o DFS preserva a integridade das réplicas. Algumas técnicas são utilizadas para garantir a consistência entre cópias, tais como: *Write Only Read Many (WORM)*, *lock* e *lease* [6, 17].

No modelo WORM, o processo de adição de novos arquivos consiste em: criar o arquivo e abri-lo para escrita; adicionar o conteúdo e fechá-lo. Após isto, o conteúdo do arquivo não pode ser alterado ou removido, apenas a adição de novos dados é permitida reabrindo o arquivo. As alterações que envolvem o conteúdo já armazenado no arquivo implica na sua exclusão e reinserção no sistema.

No modo de *lock*, é empregado um bloqueio de leitura quando o acesso é de escrita, ou bloqueio de escrita quando um arquivo está sendo acessado para leitura. Desta forma, o acesso à escrita implica na garantia de que ninguém está lendo o arquivo. Da mesma maneira, quando o arquivo for acessado para leitura, é garantido que ele não está sendo atualizado, obtendo portanto, a última versão do arquivo.

Na consistência baseada em *leasing*, é determinado um período de tempo para que um dado possa ser atualizado. Quando este tempo expira, considera-se seu conteúdo como sendo a última versão. Caso o arquivo ainda necessite de alterações, é feita uma nova alocação de tempo para uma nova atualização. Embora um arquivo possa ser periodicamente atualizado, com este modelo, quando uma solicitação de acesso é feita entre as atualizações, tem-se a garantia de que neste momento a última versão do arquivo é retornada.

A implementação de uma técnica de sincronização e consistência em um DFS, garante seu compromisso com a replicação e visa atender o conceito de tolerância a falhas.

4.2.7 Balanceamento de Carga

Segundo Benjamim et al. [6], o balanceamento de carga em um DFS é composto por dois conceitos. Um deles trata da distribuição da carga do sistema levando em consideração que servidores podem entrar e sair do DFS a qualquer tempo. Este mecanismo de balan-

ceamento deve estar disponível nos sistemas e dá suporte ao conceito de transparência e tolerância a falhas.

O outro conceito considera a capacidade de um DFS adicionar recursos no ambiente. Deste modo, o balanceamento de carga pode feito de maneira automática ou manual. Considera-se que o balanceamento é manual quando o utilizador percebe o esgotamento de recursos e, manualmente, adiciona novos servidores no ambiente.

Um DFS com balanceamento automático possui a capacidade de incluir novos recursos no sistema automaticamente. Isto significa que um ambiente pode possuir mais servidores do que o utilizado e a agregação dos recursos ociosos ocorre de acordo com a utilização. Este modelo beneficia aplicações com comportamentos não padronizados, isto é, aplicações que sofrem acessos do tipo rajada.

A inclusão de recursos no modo automático é feita por meio de uma configuração pré-determinada que leva em consideração regras e limites. Essas regras e limites são indicadores de estado do sistema (ex. espaço utilizado em disco, utilização da memória) que quando atingidos, disparam o processo de balanceamento.

4.2.8 API - *Application Programming Interface*

A interação do cliente com o DFS pode ser realizada de diversas maneiras, entre elas tem-se: CLI (*Command Line Interface*), ponto de montagem e API [6].

O acesso via CLI é feito da mesma forma como os sistemas não distribuídos, por meio de comandos (ex. Linux: *cp, rm, mv*). Outra maneira encontrada em muitos sistemas de arquivos distribuídos, é o ponto de montagem. Com essa técnica, pode-se anexar um diretório remoto de outro servidor como se este fosse uma unidade de disco local. Exemplos de como se obtém um ponto de montagem são com o comando *mount* do Linux e o ambiente FUSE[1].

A API é um conjunto de rotinas que permite a comunicação entre diferentes aplicações. Essa interface de programação oferece as funcionalidades do sistema omitindo detalhes de implementação [1]. Uma API geralmente é escrita em mais linguagens de programação do

[1] http://fuse.sourceforge.net

que a do próprio DFS. Isto porque ela tem o objetivo de facilitar a interação com aplicações externas ao ambiente. Alguns exemplos de linguagens de programação utilizadas em APIs são: C, C++, Java, PHP, Python e Ruby. Além destas, existe a API REST, que é direcionada para a integração de aplicações *web (HTTP)*.

4.3 Análise Comparativa

Nesta seção, inicialmente é apresentada uma descrição de diversos DFS propostos na literatura. Em seguida, é feita uma análise comparativa entre os sistemas considerando as definições e conceitos apresentados neste capítulo.

4.3.1 Descrição dos Sistemas de Arquivos Distribuídos

Existem na literatura trabalhos que descrevem e analisam DFS, tal como Levy e Silberschatz [23] e Thanh et al. [40]. Além destes, uma análise detalhada de sistemas de arquivos distribuídos pode ser encontrada em [6]. Este último trabalho além de conceitos, apresenta uma comparação sobre os DFS atuais.

Baseando-se nestes trabalhos, são apresentados na sequência alguns sistemas de arquivos distribuídos. A discussão é realizada como um estudo comparativo entre alguns dos DFS mais utilizados atualmente, tendo como parâmetros os conceitos apresentados na seção 4.2.

Os DFS analisados são: HDFS [17, 18], iRODS [19, 20], Ceph [41] e GlusterFS [14, 15]. Todos os sistemas avaliados são trabalhos continuados. Com isso, eles possuem algum tipo de suporte, seja comercial ou pela comunidade de código aberto.

4.3.1.1 HDFS - *Hadoop Distributed File System*

O HDFS é um sistema de arquivos distribuído de código aberto. Atualmente, ele é mantido pela *Apache Software Foundation* e está sob sua licença, denominada *Apache license 2.0*. Foi concebido para ser capaz de lidar com um grande volume de dados, na ordem dos *petabytes*.

A gestão de nomes no HDFS é feita com a técnica de indexação. Neste DFS, essa estrutura de metadados é conhecida como *inode*. Nela estão contidas todas as informações de gerência dos dados armazenados no sistema, tais como: permissão, nome, localização e data de acesso.

Para o armazenamento dos *inodes*, o HDFS utiliza um servidor central que é chamado de *NameNode*. Além deste, existe um conjunto de outros servidores que são responsáveis por armazenar os arquivos, chamados de *DataNodes*. No HDFS, todos os servidores (*NameNode, DataNodes*) são considerados como completamente conectados. Essa interligação facilita o processo de detecção de falhas, que neste sistema ocorre de forma automática. A detecção funciona com o envio e recebimento de mensagens conhecidas neste DFS como *Heartbeat*. Todos os servidores da rede são capazes de trocar estas mensagens simultaneamente.

A forma como os clientes acessam e interagem com o HDFS pode ser feita de diversas maneiras. Entre as mais conhecidas, está o ambiente de montagem FUSE. Além desta, este DFS possui APIs nas linguagens de programação C e Java. Ele conta também com a interface REST.

Em relação à replicação e localização, o HDFS implementa estes conceitos para que isto ocorra de forma automática. O processo de criação das réplicas dos arquivos armazenados nos *DataNodes* é disparado automaticamente. Com o auxílio do servidor de metadados (*NameNode*), a localização de destino das cópias de um arquivo é conhecida de modo antecipado pelo sistema. Embora no padrão do HDFS sejam três cópias, a aplicação pode determinar uma quantidade diferente de réplicas. Ainda que a replicação e sua política de localização sejam definidas automaticamente, o HDFS não fornece mecanismos para o controle exato da localidade dos arquivos.

No que diz respeito à sincronização, este DFS é o único que permite uma sincronização no modo assíncrono. Quando um cliente faz uma requisição para o sistema, ele o faz para o *NameNode*. Este, por sua vez, coordena as ações de leitura e escrita de modo a não bloquear o acesso ao arquivo. Para garantir que as cópias sejam idênticas, o HDFS implementa a consistência do tipo WORM.

A fim de prover escalabilidade, o HDFS possui uma estratégia de balanceamento considerada automática. Isto significa que este DFS é capaz de adicionar recursos ao ambiente sem a intervenção do usuário. O balanceamento é feito por meio de limites. No HDFS, o limite que dispara o processo é um valor pré-determinado entre 0 (vazio) e 1 (cheio) que corresponde ao espaço utilizado pelo armazenamento do sistema.

4.3.1.2 iRODS

Este sistema foi desenvolvido pelo grupo de pesquisa *Data Intensive Cyber Environments (DICE)*. Embora seja uma aplicação de código aberto, passou a ser comercializado como um produto e seu nome é marca registrada.

Muito similar ao HDFS no quesito arquitetura, este DFS possui um servidor centralizado responsável por armazenar os metadados, conhecido como *iCat server*. Além deste, há um conjunto de outros servidores que armazenam os arquivos. O agrupamento do servidor de metadados e os demais servidores neste DFS recebem o nome de zona. Para prover a nomeação em uma zona, o iRODS utiliza uma técnica exclusiva entre os sistemas analisados. Ele mantém os metadados em uma base de dados tradicional. Desta forma, é capaz de oferecer ao usuário a manipulação deste servidor por meio da linguagem SQL.

O acesso ao sistema pela aplicação pode ser feita com o ambiente FUSE, ou então, utilizando as APIs disponíveis nas linguagens PHP e Java. O envio de um comando para a execução no iRODS por meio das APIs é conhecida nesse sistema como a execução de uma regra (ex. acCreateUser, acDeleteUser, msiSysChksumDataObj). Com isso, o DFS ganha em flexibilidade, pois permite que regras possam ser criadas para a execução imediata ou agendadas para início futuro. Desta forma, a criação de regras pode contribuir para o controle de localidade. Isto porque podem ser criadas regras de execução periódica para reorganizar os arquivos armazenados e assim determinar sua localidade.

A detecção de falhas neste DFS é feita automaticamente. O ambiente do iRODS é completamente conectado, isto é, cada servidor tem acesso direto a todos os demais. Deste modo, todos os servidores do sistema são capazes de trocar mensagens entre si para sinalizar que estão ativos.

Embora a conectividade seja completa no iRODS, o que auxilia no processo de replicação, por padrão, ela é não ativa. Este processo deve ser iniciado manualmente pelo utilizador do sistema, indicando também a forma como os arquivos devem ser distribuídos. Além disso, este DFS não permite que o balanceamento seja feito de modo automático. Toda adição de recursos no ambiente deve ser provida manualmente pelo usuário do sistema.

Ainda que a replicação seja manual, como o iRODS provê a criação e agendamento de regras para execução futura, pode-se criar uma regra para a execução da replicação, tornando-a virtualmente automatizada.

Em termos de sincronização, no iRODS ela é feita de modo síncrono. Dessa forma, o sistema garante que um dado só é entregue para a aplicação quando todas as cópias estiverem atualizadas. Para garantir a consistência entre as réplicas, o DFS utiliza o mecanismo de WORM.

4.3.1.3 Ceph

O Ceph é um sistema de arquivos distribuído de código aberto sob a licença LGPL. Seu desenvolvimento foi iniciado na Universidade de Califórnia, Santa Cruz (EUA) pelo então aluno Sage A. Weil. Ele continuou se dedicando ao projeto até a conclusão de seu doutorado na mesma universidade. Posteriormente, ele criou uma empresa chamada Inktank Storage para comercializar o suporte ao DFS. A empresa foi adquirida e atualmente o Ceph pertence à Red Hat[2].

Este DFS possui uma flexibilidade em relação aos demais porque pode ser utilizado como um sistema de objetos, blocos ou arquivos distribuídos. Isto é possível graças a implementação do RADOS [43]. O funcionamento do RADOS consiste em criar uma plataforma de armazenamento totalmente distribuída que dispensa a figura de um servidor central para gerenciar os metadados. Esta camada de armazenamento provida via RADOS é chamada de *Ceph Storage Cluster (CSC).*

O acesso ao *Ceph Storage Cluster* pode ser feito para o armazenamento de blocos com a

[2]http://www.redhat.com

utilização da biblioteca chamada **librbd**, para armazenar objetos com a interface REST e como sistema de arquivos tradicional, com o ambiente FUSE. Além disso, o Ceph permite acesso direto ao CSC com a utilização da biblioteca **librados**. Esta biblioteca possibilita que qualquer aplicação possa utilizar o sistema. Ela está disponível nas linguagens C, C++, Java, Python, Ruby e PHP.

A fim de prover o conceito de transparência e tolerância a falhas, o Ceph cria uma abstração entre o armazenamento físico e o lógico. Essa abstração é o equivalente a uma unidade de armazenamento e é chamada de *pool*. Os *pools* são mapeados pelo algoritmo CRUSH [42] e o resultado é conhecido como CRUSH *Map*. Neste mapa, constam as informações sobre a associação que existe entre os *pools* e seu armazenamento nos servidores do ambiente. Desta forma, a replicação e localização das cópias ocorre automaticamente. O Ceph baseia-se no CRUSH *Map* para determinar quando, quantas e onde as cópias devem ser mantidas. Além disso, por meio da integração das API com o CRUSH *Map*, o Ceph fornece controle da localidade dos dados.

Com a utilização deste recurso, pode-se organizar o armazenamento dos arquivos de acordo com sua utilização. Para isso, é criado no arquivo do CRUSH *Map* um conjunto de regras. Estas regras são associadas às unidades físicas de armazenamento (Disco, Máquina, Rack), por exemplo: DiscoA = *regra1*; DiscoB = *regra2*. Assim, adiciona-se também no CRUSH *Map* a associação entre a regra e o *pool*, exemplo: *pool*1 = *regra1*. Logo, pode-se determinar que todos os dados pertencentes ao *pool*1 estão armazenados no DiscoA. Desta forma, obtém-se o controle de localidade dos dados permitindo que otimizações baseadas em consultas possam ser implementadas pela aplicação.

A Figura 4.1 apresenta a associação entre unidades físicas, regras e *pools* no DFS Ceph.

Ainda que seja possível determinar a quantidade de réplicas e em quais servidores elas serão armazenadas, o Ceph, por meio do algoritmo CRUSH, atribui para cada servidor a função de réplica primária de um conjunto de arquivos. Desta forma, quando um arquivo é solicitado, o DFS busca preferencialmente este arquivo no servidor considerado como réplica primária. Assim, para garantir que não haja comunicação de rede entre os

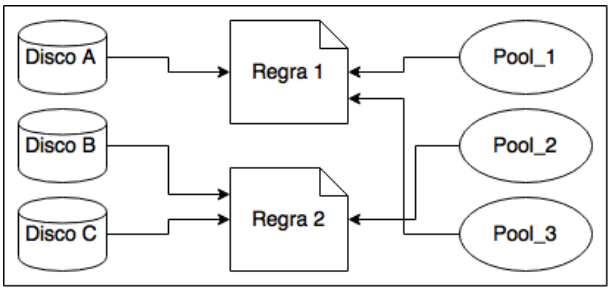

Figura 4.1: Associação entre unidades físicas, regras e *pools* no DFS Ceph.

servidores do ambiente, é preciso identificar qual servidor é a réplica primária do arquivo que está sendo requisitado.

Para a manipulação dos nomes de arquivos no sistema, o Ceph utiliza o mesmo algoritmo CRUSH. Este algoritmo pode ser aplicado por qualquer servidor da rede a fim de localizar um determinado arquivo no ambiente. Seu funcionamento consiste em calcular um identificador único baseado no *pool* que irá armazenar o arquivo. Em seguida, o algoritmo concatena este identificador único com o nome textual do arquivo fornecido pela aplicação, criando assim um nome global único.

A detecção de falhas no Ceph ocorre de forma automática. Como no sistemas anteriormente descritos, existe uma troca de mensagens periódica entre os servidores da rede para que cada servidor possa saber se os demais estão ativos.

Em relação à sincronização e consistência, o Ceph executa a sincronização sobre as réplicas dos arquivos de modo síncrono. Isto significa que o DFS retorna estado de sucesso para a aplicação somente quando todas as cópias do arquivo são inseridas ou atualizadas. Para garantir a consistência entre as réplicas, o Ceph implementa o mecanismo de *lock*. Quando um arquivo está sendo acessado para escrita, cria-se um bloqueio até que este seja atualizado. O *lock* geralmente ocorre no momento da abertura de um arquivo, e liberado após seu fechamento.

Embora o Ceph permita o balanceamento de carga entre os servidores ativos, com o auxílio do CRUSH *Map*, a inclusão de novos servidores no ambiente deve ser feita manualmente pelo usuário. O Ceph fornece um conjunto de comandos para se obter o estado do sistema. Com essas informações, o utilizador pode identificar a necessidade de

adicionar ou não novos recursos no ambiente.

4.3.1.4 GlusterFS

O GlusterFS é um sistema de arquivos distribuído de código aberto sob licença GPL e é desenvolvido pelo *Gluster Core Team*. Este sistema está acessível para o usuário apenas com o ambiente FUSE. Com este mecanismo de acesso, o GlusterFS ainda não possui uma funcionalidade para que a aplicação controle com precisão a localidade dos arquivos armazenados.

Sob vários aspectos o GlusterFS é muito similar ao Ceph. Os mais notáveis são em relação a arquitetura e nomeação. A começar pela arquitetura, o GlusterFS utiliza-se da mesma abordagem que o Ceph. Ele distribui os arquivos e metadados como uma entidade única, não necessitando de uma estrutura de metadados centralizada. Para localizar os arquivos e prover nomeação, ele também faz uso de um algoritmo chamado de EHA (*Elastic Hashing Algorithm*) [15]. Este algoritmo se utiliza de uma função *hash* baseada no algoritmo de Davies-Meyer. O EHA obtém um nome global único por meio do resultado da função *hash* aplicada sobre o nome textual do arquivo.

O GlusterFS é capaz de identificar falhas de modo automático. Assim como no Ceph, utiliza-se a troca de mensagens pela rede para se obter o estado de cada servidor. Além disso, o GlusterFS não é capaz de incluir novos recursos automaticamente. O processo de balanceamento de carga deve ser iniciado pelo utilizador do sistema.

Em relação à replicação, este DFS possui uma característica que o difere dos demais. A exemplo do iRODS, a replicação deve ser iniciada manualmente pelo utilizador do sistema, entretanto, não é feita por arquivos, mas por volumes inteiros. Um volume neste sistema pode representar um único disco ou então várias partições.

No GlusterFS, o processo de sincronização entre réplicas de arquivos é realizado de modo síncrono. Quando o processo de replicação é iniciado, os arquivos que estão em um volume que está sendo replicado ficam bloqueados para acesso até que o processo seja concluído. Para preservar a consistência entre os volumes, o sistema implementa a consistência do tipo WORM.

4.3.2 Análise

A Tabela 4.1 apresenta um resumo da análise comparativa entre os DFS.

	HDFS	**iRODS**	**Ceph**	**GlusterFS**
Licença	Apache 2.0	*Copyright*	LGPL	GPL
Arquitetura de Gerência de Metadados	Centralizada	Centralizada	Distribuída	Distribuída
API	C, Java e REST	Java e PHP	REST, C, C++, Java, PHP, Ruby e Python	Não
Nomeação	Índices	Base de Dados	CRUSH	EHA
Detecção de Falhas	Automática	Automática	Automática	Automática
Replicação e Política de Localização	Automática	Manual	Automática	Manual
Controle de Localidade	Não	Sim	Sim	Não
Sincronização / Consistência	Assíncrona / WORM	Síncrona / WORM	Síncrona / Lock	Síncrona / WORM
Balanceamento de Carga	Automática	Manual	Manual	Manual

Tabela 4.1: Tabela comparativa entre os sistemas de arquivos distribuídos.

Baseando-se na análise do sistemas de arquivos distribuídos, são notáveis algumas semelhanças e diferenças entre os DFS. A principal similaridade fica por conta da replicação. Muito embora nem todos a façam de forma automática, disponibilizar este recurso é vantajoso para o sistema, uma vez que essa técnica está associada à tolerância a falhas e portanto, à disponibilidade. Em termos de diferenças, a mais notável é em relação à arquitetura de cada sistema. O HDFS e iRODS possuem um elemento centralizador que é responsável pelos metadados. Isto implica em acessos ao servidor central para a execução das operações de criação, leitura, escrita e exclusão. Neste quesito, o GlusterFS e o Ceph se diferenciam dos demais pois possuem arquitetura totalmente distribuída, e, desta forma, permitem que qualquer servidor participante do sistema seja capaz de localizar os arquivos por meio do emprego de algoritmos EHA [15] e CRUSH [42] respectivamente.

Nestes sistemas, GlusterFS e Ceph, percebe-se que tanto a detecção de falhas quanto o balanceamento de carga ocorre de maneira similar. Entretanto, ambos possuem diferenças que podem ser mensuradas por três aspectos: replicação, API e controle de localidade. Para a replicação, nota-se que no Ceph ela é feita de forma automática e por arquivos. Já

no GlusterFS, é feita manualmente e por volumes inteiros. No caso da API, que é uma peça chave na integração entre aplicação e DFS, percebe-se que o GlusterFS fornece acesso somente por ponto de montagem. Em contrapartida, além deste mecanismo, o Ceph provê acesso também por API com diferentes linguagens de programação. Associado ainda às APIs, outro recurso que difere estes DFS é o controle de localidade. Isto acontece porque em um ambiente com ponto de montagem não é possível determinar a localização de forma explícita e exata de um arquivo. Sob estes aspectos, constata-se que o sistema de arquivos distribuídos Ceph demonstra-se uma escolha adequada para a continuidade deste trabalho.

CAPÍTULO 5

ARQUITETURA DASFLOW

Este capítulo descreve os conceitos e funcionamento da arquitetura DASFlow. A arquitetura DASFlow é uma solução proposta para prover escalabilidade na atividade de monitoramento de rede. Para isso, esta arquitetura é planejada e aplicada na ferramenta de monitoramento de rede NfSen/Nfdump. Inicialmente será apresentada a visão geral da arquitetura. Em seguida, é apresentado o arquivo de configuração fornecido como entrada da arquitetura e utilizado para estabelecer a organização dos dados armazenados. Após, o modelo de dados utilizado pelo metadados é descrito em detalhes. Posteriormente, são descritos os nodos cliente e servidor com seus componentes e funcionamento. Por fim, o capítulo descreve o processo realizado pela arquitetura para processar as consultas de modo distribuído.

5.1 Visão Geral

A arquitetura DASFlow foi criada a fim de prover escalabilidade para as tarefas de coleta, armazenamento e processamento do monitoramento de rede aplicada à ferramenta NfSen/Nfdump. A Figura 5.1 ilustra a arquitetura DASFlow, incluindo os nodos, seus componentes e a ferramenta NfSen/Nfdump.

A arquitetura considera um conjunto de nodos. Cada nodo desempenha o papel de cliente, servidor ou ambos. O nodo cliente tem como componentes principais os módulos StoreDAS-Cliente e QueryDAS-Cliente, enquanto o StoreDAS-Servidor e QueryDAS-Servidor são os módulos que compõem o nodo servidor. O módulo QueryDAS-Cliente é responsável por receber consultas submetidas pelo administrador de rede, distribuí-las pelos nodos servidores, processar os resultados parciais enviados pelos servidores e retornar o resultado da consulta. Nos nodos servidores, o módulo QueryDAS-Servidor é responsável pelo processamento parcial da consulta sobre os dados armazenados no próprio servidor.

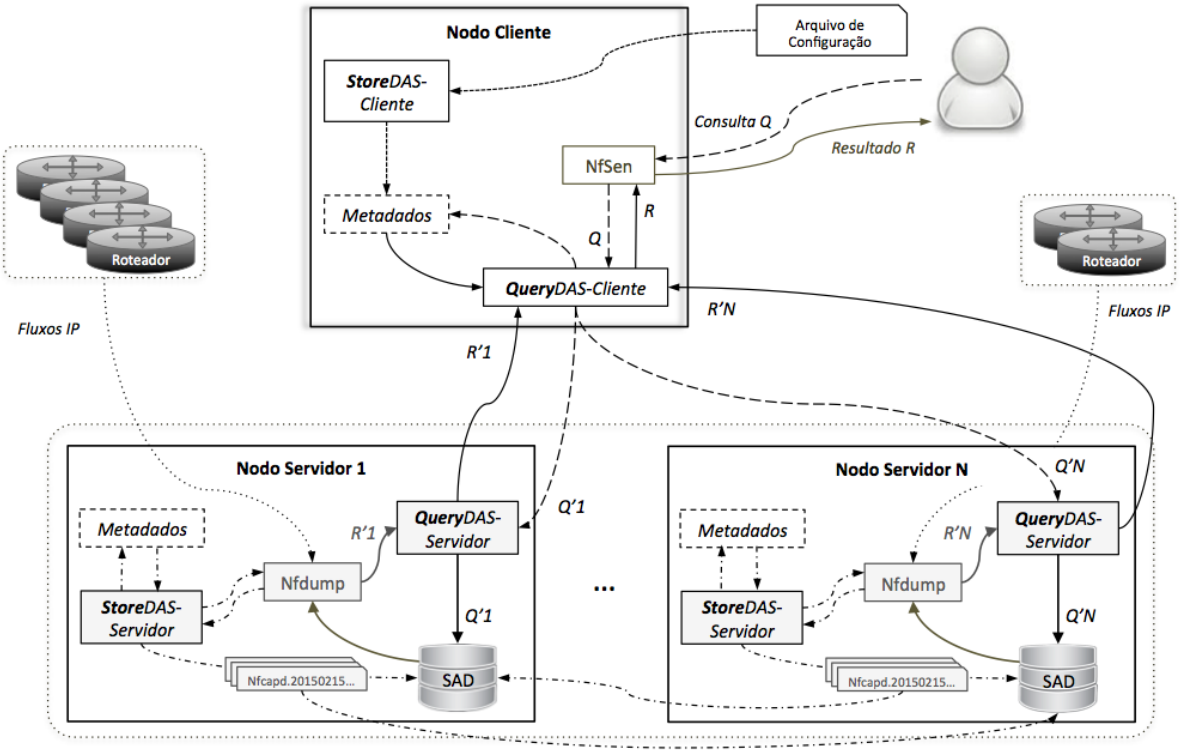

Figura 5.1: Visão geral da arquitetura distribuída DASFlow.

O componente que mantém as informações de distribuição e replicação dos dados de monitoramento é o módulo de Metadados, que são gerenciadas pelo StoreDAS-Cliente e StoreDAS-Servidor nos nodos cliente e servidor, respectivamente. Os metadados associam conjuntos de arquivos com dados de um intervalo de tempo, a um conjunto de nodos servidores a partir de um arquivo de configuração dado como entrada. O arquivo de configuração é fornecido pelo administrador para indicar como a organização dos arquivos armazenados pelo sistema deve ser realizada. Neste arquivo pode-se, por exemplo, determinar que o conjunto de dados das últimas 24 horas sejam replicados em todos os servidores do ambiente (ex: Servidor1, Servidor2, Servidor3 e Servidor4). Estes arquivos possuem dados recentes de monitoramento, ou também chamados de dados *quentes*. Além disso, o arquivo de configuração pode determinar que apenas poucas réplicas dos dados mais antigos, chamados pela arquitetura de dados *frios*, sejam mantidos somente em alguns servidores do ambiente (ex: Servidor1 e Servidor3). Um conjunto de servidores que armazena as réplicas de um mesmo conjunto de dados frios é chamado pela arquitetura de fila. O número de réplicas para os conjuntos de dados e a quantidade de filas

existentes no ambiente são ajustadas pelo administrador no arquivo de configuração.

É importante observar que os metadados são replicados em todos os nodos do sistema. Assim, o servidor que recebe os dados de monitoramento sabe para onde os arquivos devem ser encaminhados para armazenamento sem consultar um nodo cliente. A arquitetura prevê a existência de diversos servidores, associados a um ou mais roteadores. Desta forma, à medida que o tráfego aumenta, é possível adicionar novos servidores ao sistema, ou seja, prover escalabilidade de coleta.

Na sequência, são descritos em detalhes o arquivo de configuração, os metadados e os nodos cliente e servidor.

5.2 Arquivo de configuração

O módulo StoreDAS-Cliente recebe como entrada um arquivo de configuração criado pelo administrador de rede. Neste arquivo, constam informações sobre a quantidade de servidores no ambiente além da distribuição e organização dos arquivos armazenados. Com isso, o módulo pode organizar de forma automatizada o armazenamento dos arquivos e alimentar o módulo de Metadados. Isto é, este módulo mantém atualizadas as informações sobre a distribuição dos dados para que o módulo QueryDAS-Cliente possa localizar com precisão em qual nodo servidor do ambiente os arquivos estão armazenados.

O arquivo de configuração permite que o administrador de rede indique o período em que os arquivos serão considerados como dados quentes e, armazenados em todos os servidores (réplica total). Adicionalmente, o arquivo de configuração fornece informações para que os metadados mantenham o registro sobre a distribuição dos dados antigos, ditos dados frios. Os dados quentes e frios são utilizados geralmente para atender, respectivamente, consultas investigativas e gerenciais, conforme descrito anteriormente na seção 2.2.5. A Figura 5.2 mostra um exemplo do arquivo de configuração.

A arquitetura considera que os servidores são configurados pelo administrador para que possam se comunicar entre si utilizando o nome de cada servidor. A primeira configuração do sistema deve ter como número de série 0001 (linha 4), configurações posteriores devem incrementar o número de série do arquivo. Sem isso, as modificações são ignoradas pelo

```
01 # Arquivo de configuração do Metadados utilizado pelo módulo StoreDAS-Cliente
02 # Número de série deve ser alterado a cada modificação
03 # Número de série:
04 0001
05 #
06 Seção 1 - Filas, Servidores e Réplicas
07 # Nesta seção são fornecidas as quantidades e nomes das filas, e,
08 # os nomes dos servidores em suas respectivas filas
09 # O número de réplicas para cada fila é definido pela quantidade de servidores inseridos em cada fila
10 # Ex: Fila01 com os servidores A e C (2 réplicas)
11 #
12 Fila01: Servidor-A, Servidor-C
13 Fila02: Servidor-B, Servidor-D
14 #
15 ###*###
16 # Seção 2 - dadosQuentes
17 # Nesta seção é definido o tempo que os dados serão considerados como recentes (dados quentes)
18 # O valor fornecido deve ser fornecido em (H)oras. Ex: 24, 48, 72
19 # Valor padrão: 672 Horas (28 Dias)
20 #
21 672
22 #
23 ###*###
24 # Seção 3 - dadosFrios - Regras
25 # Nesta seção do arquivo são definidas as regras para o armazenamento dos dados antigos (dados frios)
26 #
27 # Subseção 3.A - Algoritmo de Escalonamento
28 # Nesta subseção pode-se escolher o algoritmo que irá escalonar os dados armazenados
29 # As opções são:
30 # RR (Round Robin): Este algoritmo faz o balanceamento igualitário entre as filas existentes
31 # Melhor uso: servidores com capacidade de armazenamento homogênea
32 # MFS (Most Free Space, Maior Espaço Livre): Este algoritmo baseia-se no espaço
33 # disponível de cada fila para determinar onde os dados serão armazenados
34 # Melhor uso: servidores de capacidade heterogênea ou após a adição de novos servidores
35 # Valor padrão: RR
36 #
37 RR
38 #
39 # Subseção 3.B - Intervalo de agrupamento - (H)oras
40 # Esta subseção determina o intervalo para que os dados sejam considerados um subconjunto
41 # Este valor é utilizado, por exemplo, pelo algoritmo RR como indicador de troca de fila
42 # Valor padrão: 6 Horas
43 #
44 6
45 #
```

Figura 5.2: Arquivo de configuração do módulo de Metadados.

módulo StoreDAS-Cliente.

Na linha 6 do arquivo, é descrita a primeira seção de configuração a respeito da organização dos servidores no ambiente. Para isso, o administrador agrupa os servidores em conjuntos que são identificados pela arquitetura como filas (linhas 12 e 13). Para cada fila, são indicados um ou mais servidores, com o máximo de três, que serão responsáveis por armazenar as réplicas de um mesmo conjunto de dados frios. Logo, pode-se dizer que conjuntos de arquivos com dados de um mesmo intervalo de tempo são armazenados em filas.

Na linha 21, o administrador informa o período em que os arquivos são considerados como dados quentes. Para os arquivos considerados quentes, a arquitetura não aplica o agrupamento dos dados por fila, e sim, distribui uma réplica de cada arquivo para cada servidor. A organização dos arquivos com dados quentes é realizada desta maneira para melhorar o desempenho das consultas, uma vez que os dados recentes tendem a ser acessados com maior frequência. O valor fornecido neste campo deve ser expresso em horas.

A partir da linha 24, são informados os valores para a organização dos arquivos que possuem dados frios e que serão associadas às filas. Inicialmente, o administrador indica o algoritmo utilizado para o balanceamento entre as filas. Isto é, como é realizada a distribuição e armazenamento dos conjuntos de arquivos entre os conjuntos de servidores ativos do sistema (filas). Os valores possíveis para este campo são: RR (*Round Robin*) e MFS (*Most Free Space*).

Na linha 44, é fornecido o intervalo de tempo utilizado para criar os conjuntos de dados e indicar o período de rotacionamento entre as filas. Este valor deve ser expresso em horas. Para exemplificar, considere o valor padrão de 6 horas para os arquivos de um dia inteiro. Este valor indica que os arquivos compreendidos em um intervalo de 6 horas (00:00h - 05:55h) são armazenados em todos os servidores que compõem a Fila01 e formam um mesmo conjunto de dados. Os arquivos do próximo período (06:00h - 11:55h) são armazenados na próxima fila (Fila02) e do período de 12:00 - 17:55 voltam a ser armazenados na Fila01, de acordo com o escalonamento *Round Robin*.

A Figura 5.3 mostra como se comporta a organização dos servidores e como é realizada a distribuição dos dados considerando o arquivo de configuração da Figura 5.2.

Pode-se observar na Figura 5.3 como é criada a organização dos servidores em filas e o número de réplicas para cada fila. Neste exemplo, foram considerados 4 servidores, sendo eles distribuídos entre duas filas (Fila01 e Fila02), e cada fila por sua vez possui 2 réplicas.

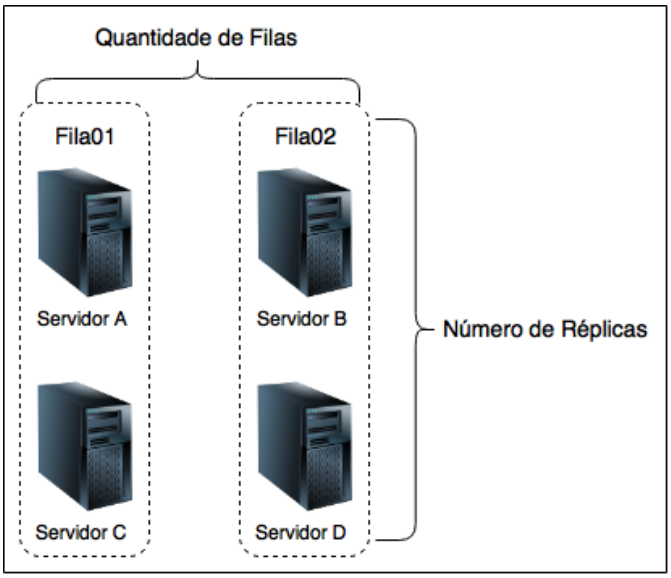

Figura 5.3: Exemplo de distribuição dos dados utilizando filas.

5.3 Metadados

O módulo de Metadados é utilizado pelos módulos StoreDAS e QueryDAS para armazenar e recuperar, respectivamente, as informações sobre a distribuição e localização dos dados de monitoramento na arquitetura. Para isso, o metadados utiliza o modelo de dados apresentado na Figura 5.4. O modelo de dados é composto de 3 tabelas identificadas como: T_Dados, T_Filas e T_Config.

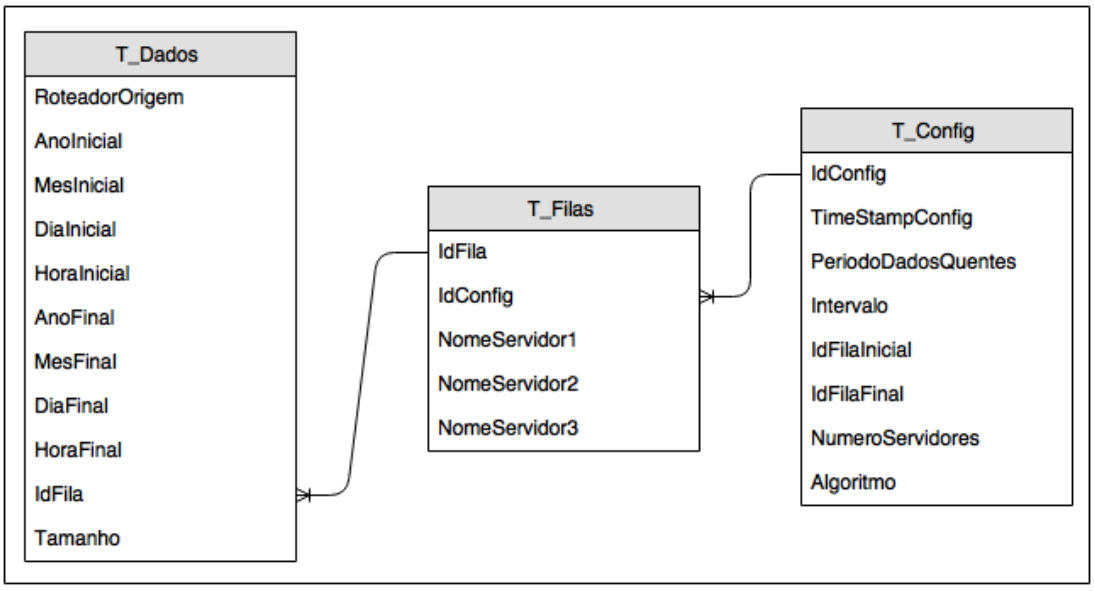

Figura 5.4: Modelo de dados utilizado pelo módulo Metadados.

As informações armazenadas no metadados são obtidas a partir das configurações da ferramenta NfSen/Nfdump associadas com as informações fornecidas pelo administrador por meio do arquivo de configuração. Adicionalmente, são extraídas informações relacionadas ao período em que os conjuntos de arquivos são armazenados. Desta forma, o módulo de Metadados permite localizar com precisão em quais servidores determinados conjuntos de dados estão armazenados.

Para exemplificar, considere que os arquivos do dia 12/03/2015 passaram de quentes para frios e a arquitetura irá organizá-los no ambiente e alimentar o metadados. Os valores adicionados no metadados são exemplificados na tabela T_Dados, conforme apresentado em destaque na Figura 5.5. Pode-se observar que o conjunto de arquivos que contém os dados do dia 12/03 foram armazenados no conjunto de servidores A e C, ou também pode-se dizer que estão armazenados na Fila01. Esta informação é obtida por meio do campo $IdFila$ que é o identificador da fila na tabela T_Filas.

A Figura 5.5 apresenta exemplos dos valores utilizados pelas tabelas T_Dados, T_Filas e T_Config.

A descrição de cada tabela com seus respectivos valores são apresentados em detalhes a seguir.

5.3.1 Tabela T_Config

Esta tabela é preenchida com os valores obtidos a partir do arquivo de configuração. Com isso, ela é utilizada para completar os valores das tabelas T_Filas e T_Dados. Os valores de cada campo da tabela são descritos a seguir.

IdConfig: Este campo é preenchido com o número de série obtido no arquivo de configuração. O valor deste campo é utilizado pela tabela T_Filas quando ocorrem mudanças em relação à quantidade e distribuição dos servidores no ambiente e que alteram as estruturas das filas.

TimeStampConfig: Este campo armazena o dia e a hora em que houve alteração no arquivo de configuração. Os valores deste campo ajudam a manter um histórico de versões de todas as alterações feitas na arquitetura pelo administrador.

T_Config

IdConfig	TimeStamp Config	PeriodoDados Quentes	Intervalo	IdFila Inicial	IdFila Final	Numero Servidores	Algoritmo
1	15/09/2015-15:00	672	1440	1	2	4	RR

T_Filas

IdFila	IdConfig	NomeServidor1	NomeServidor2	NomeServidor3
1	1	Servidor-A	Servidor-C	
2	1	Servidor-B	Servidor-D	

T_Dados

Roteador Origem	Ano Inicial	Mes Inicial	Dia Inicial	Hora Inicial	Ano Final	Mes Final	Dia Final	Hora Final	IdFila	Tamanho
pop-pr	2015	03	10	00	2015	03	10	23	1	19500
pop-pr	2015	03	11	00	2015	03	11	23	2	20000
pop-pr	2015	03	12	00	2015	03	12	23	1	20400

Figura 5.5: Exemplos das tabelas T_Dados, T_Filas e T_Config utilizadas pelo módulo Metadados.

PeriodoDadosQuentes: O valor deste campo corresponde ao período em que os dados são considerados quentes, ou seja, estão armazenados em todos os nodos servidores. Este valor é lido pelos módulos StoreDAS-Cliente e QueryDAS-Cliente a cada modificação da tabela T_Config para que eles possam realizar as funções de armazenamento e processamento, respectivamente.

Intervalo: Este campo armazena o período utilizado para o rotacionamento entre as filas, e é expresso em minutos. Isto é, um conjunto de dados compreendidos no intervalo de tempo são agrupados e armazenados na mesma fila. O valor 1.440 da tabela no exemplo indica que os arquivos são agrupados em períodos de 1 dia, logo, os arquivos do dia 12 estão armazenados na Fila01, os arquivos do dia 11 na Fila02 e assim por diante.

IdFilaInicial: Este campo contém o identificador da primeira fila em ativa no ambiente.

IdFilaFinal: Este campo contém o identificador da última fila em ativa no ambiente.

Assim, pode-se subtrair o valor de identificação da última fila pelo valor da primeira o obtém-se a quantidade de filas ativas no sistema.

NumeroServidores: A quantidade total de servidores no ambiente é armazenada neste campo. Ele é utilizado pelo módulo QueryDAS-Cliente para o processamento das consultas distribuídas.

Algoritmo: Neste campo é armazenado o algoritmo utilizado para o balanceamento entre as filas.

5.3.2 Tabela T_Filas

Esta tabela mantém informações sobre as filas utilizadas pela arquitetura. Ela é utilizada pela tabela T_Dados para associar os conjuntos de dados às filas de armazenamento.

IdFila: Este campo é um valor auto incrementado para cada alteração na estrutura das filas, indicada pelo administrador de rede per meio do arquivo de configuração. Os valores deste campo permitem manter o histórico de alterações nas composições de cada fila que foi criada no ambiente.

IdConfig: Este campo é obtido a partir da tabela T_Config para estabelecer a relação entre as filas e a estrutura de armazenamento criada pelo administrador por meio do arquivo de configuração. Isto é, identificar o momento em que houve alteração e como deve ser o comportamento dos dados armazenados nas filas (intervalos, algoritmo, dados quentes).

NomeServidor1: Cada linha da tabela T_Filas em seu campo $NomeServidor1$, bem como os campos $NomeServidor2$ e $NomeServidor3$, armazenam os nomes dos servidores que compõem suas respectivas filas identificadas pelo campo $IdFila$.

5.3.3 Tabela T_Dados

A tabela T_Dados armazena informações sobre a localização de armazenamento de cada conjunto de dados que pertencem a um mesmo período de tempo. Isto é, para cada conjunto de dados armazenados em uma fila é inserida uma linha nesta tabela. Além disso, ela também mantém o registro sobre a origem do tráfego relacionado ao período

juntamente com o tamanho dos dados. Os campos da tabela e seus respectivos valores são descritos a seguir.

RoteadorOrigem: Neste campo é fornecido a identificação do roteador ao qual o tráfego pertence. Esta identificação é obtida a partir da própria ferramenta NfSen/Nfdump.

AnoInicial: Este campo é preenchido de acordo com o período de tempo que se deseja armazenar as informações no metadados. Assim como os campos $MesInicial$, $DiaInicial$ e $HoraInicial$, o valor $AnoInicial$ é obtido a partir de uma parte do nome do primeiro arquivo relacionado ao período. Neste caso o arquivo `nfcapd.201503120000` é utilizado.

MesInicial: Este campo é preenchido com o mês inicial em que o primeiro arquivo do conjunto de dados pertence. No exemplo o valor utilizado é 03.

DiaInicial: Assim como o campo $MesInicial$, o dia inicial corresponde ao primeiro dia do conjunto de dados armazenados, neste caso dia 12.

HoraInicial: Neste campo indica-se a hora do primeiro arquivo relacionado ao período, que para o exemplo mostrado o valor é 00.

AnoFinal: Assim como o campo $AnoInicial$, o valor $AnoFinal$ é obtido a partir de uma parte do nome do último arquivo relacionado ao período. Neste caso o arquivo `nfcapd.201503122355` é utilizado.

MesFinal: Este campo é preenchido com o mês final em que o último arquivo do conjunto de dados pertence. No exemplo o valor utilizado 03 é o mesmo do campo $MesInicial$.

DiaFinal: O valor do dia final corresponde ao dia do último arquivo do intervalo de tempo utilizado para criar o conjunto de dados. Neste exemplo, o intervalo para o agrupamento dos arquivos é equivalente a um dia, assim, o valor do primeiro e do último arquivo são iguais, logo, o valor 12 é utilizado.

HoraFinal: O campo $HoraFinal$ é fornecido de acordo com a hora indicada no nome do último arquivo do período armazenado. Para o exemplo citado, o valor da hora no último arquivo é 23. De modo similar, se o último arquivo fosse `nfcapd.201503121955`, a hora final seria preenchida com o valor 19.

IdFila: Este campo indica em qual fila o conjunto de arquivos é armazenado. Este valor corresponde a um identificador único da fila que é obtido a partir da tabela T_Filas.

Tamanho: Este campo é preenchido obtendo-se o tamanho total dos arquivos, em *Megabytes*, do período armazenado. Ele é utilizado para dividir os períodos das consultas de modo a balancear de forma justa o volume de dados a ser processado entre os servidores.

5.4 Nodo Cliente e Nodo Servidor

Esta seção apresenta os nodos que compõem a arquitetura DASFlow. Isto é, descreve em detalhes o nodo cliente e nodo servidor com seus componentes e funcionamento.

5.4.1 Nodo cliente

Um nodo cliente possui quatro componentes: a ferramenta NfSen, os módulos de Metadados, StoreDAS-Cliente e QueryDAS-Cliente. O módulo de Metadados foi descrito anteriormente na seção 5.3. Os demais componentes e seu funcionamento são apresentados a seguir.

Ferramenta NfSen: Este componente é a interface de comunicação entre o administrador de rede e a arquitetura. Ele é responsável por receber a consulta sobre os dados de monitoramento e apresentar os resultados. É importante observar que para um administrador que já utiliza a ferramenta NfSen/Nfdump, não há nenhuma alteração na interface com a qual ele já está habituado, sendo as questões de escalabilidade tratadas internamente pela arquitetura.

StoreDAS-Cliente: Este é o componente que recebe o arquivo de configurações que determina a distribuição e replicação dos arquivos e atualiza os metadados em todos os nodos no sistema. O StoreDAS-cliente também é responsável por monitorar o tempo de vida dos arquivos armazenados em cada servidor. Assim, dados que eram considerados quentes e passaram a ser frios podem ser removidos dos servidores que não são os responsáveis por manter os dados históricos daquele intervalo de tempo.

QueryDAS-Cliente: Este módulo coordena o processamento de consultas. Para

isso, ele acessa os metadados para encontrar os nodos servidores que armazenam os dados envolvidos em uma consulta (Q). Com isso, o módulo faz o particionamento do conjunto de dados da consulta Q em subconjuntos de dados. Estes subconjuntos de dados são processados pelos nodos servidores e os resultados parciais (R') são devolvidos para o nodo cliente. Os resultados parciais são armazenados em arquivos temporários criados por meio da execução do Nfdump em cada nodo servidor. O módulo QueryDAS-Cliente executa então o processamento final sobre as saídas parciais para gerar o resultado completo (R). O resultado final é gerado pela execução da consulta (Q) sobre os resultados parciais (R') recebidos dos nodos servidores. Uma descrição detalhada do processamento de consultas realizado pelo módulo QueryDAS-Cliente é apresentada na seção 5.5.

5.4.2 Nodo servidor

O armazenamento dos arquivos de monitoramento, bem como o processamento de consultas sobre estes arquivos são realizados pelos nodos servidores. Os componentes que atuam nestes nodos são apresentados a seguir.

Nfdump: Este componente faz parte da ferramenta NfSen/Nfdump e é responsável por realizar a coleta e processamento parcial das consultas.

Sistema de arquivos distribuído (SAD): Este elemento está disponível em todos os nodos servidores formando um ambiente de armazenamento distribuído. Ele é responsável por permitir o acesso aos arquivos a partir de qualquer nodo da arquitetura, através de um espaço de nomes de arquivos comum.

StoreDAS-Servidor: Este módulo é responsável por prover escalabilidade de coleta à arquitetura, atuando como um intermediador entre a coleta realizada pelo Nfdump e o sistema de arquivos distribuído. Assim que o Nfdump gera o arquivo contendo os fluxos IP, o StoreDAS-Servidor adiciona ao seu nome informações de identificação, isto é, a qual roteador o tráfego pertence. Este arquivo é então armazenado em um ou mais servidores de armazenamento, de acordo com as informações obtidas dos metadados.

QueryDAS-Servidor: Este módulo é responsável por receber a consulta Q enviada pelo QueryDAS-Cliente para ser processada sobre um subconjunto de dados armazena-

dos no próprio servidor. Após esta etapa, o módulo solicita ao Nfdump que execute o processamento sobre os arquivos recuperados do SAD. Neste passo, o Nfdump utiliza o parâmetro '-w' para criar uma saída temporária que será armazenada em um arquivo. Esta saída possui o formato de entrada do Nfdump, logo, é possível reutilizar este resultado para novas consultas. Ao concluir o processamento, o resultado parcial (R') é enviado ao nodo cliente. O QueryDAS-Servidor e QueryDAS-Cliente são os componentes responsáveis por prover a escalabilidade de processamento de consultas.

Um dos requisitos esperados da arquitetura DASFlow é a portabilidade e continuidade do projeto. Assim, foi determinado que a implementação da solução seria na forma de módulos independentes da ferramenta original. Deste modo, eles podem ser anexados à ferramenta NfSen exigindo alterações mínimas no código original. Adicionalmente, a arquitetura proposta visa atender de modo escalável e na mesma solução os três aspectos do monitoramento. A coleta e o armazenamento são atendidos pelo sistema de arquivos distribuído e pelos módulos StoreDAS. Para a escalabilidade de processamento, os módulos QueryDAS dividem o conjunto de dados de uma consulta (Q) em subconjuntos de dados para obter a cooperação de cada nodo servidor na execução da consulta completa.

5.5 Processamento de Consultas na Arquitetura DASFlow

As consultas realizadas pelo administrador de rede envolvem um período (P) de tempo do qual se deseja obter informações sobre o tráfego de rede conforme descrito no capítulo 2. Os arquivos solicitados na consulta são armazenados pela arquitetura seguindo o conceito de dados quentes e frios, descritos anteriormente. Quando uma consulta Q é submetida, o módulo QueryDAS-Cliente deve identificar se, para responder a consulta, o período solicitado corresponde aos dados quentes, frios ou ambos. Após identificar os dados envolvidos no período P da consulta, o módulo divide este período para formar subconjuntos de dados para que os nodos servidores as processem.

O Algoritmo 1 descreve o processo utilizado pelo módulo QueryDAS-Cliente para particionar um conjunto de arquivos solicitados em uma consulta (Q) em subconjuntos de arquivos.

Algorithm 1: particionamento de arquivos de entrada entre os servidores S1, S2, ..., Sn

Entrada: periodoInicial, periodoFinal
Saída: para cada servidor Si, um conjunto de arquivos $F[Si]$

```
1  begin
2      para cada servidor Si: F[Si] ← {};
3      periodo ← periodoFinal;
4      servidor ← S1;
5      while periodo >= horaAtual - T_Config.PeriodoDadosQuentes do
6          F[servidor] ← F[servidor]U{"nfcapd." + periodo};
7          periodo ← periodo - 5 minutos;
8          servidor ← próximo servidor Si + 1;
9      end
10     intervalo ← obtém linha da tabela T_Dados que contém o periodo;
11     while periodo >= periodoInicial do
12         intFim ← intervalo.AnoFinal + intervalo.MesFinal +
           intervalo.DiaFinal + intervalo.HoraFinal;
13         intIni ← intervalo.AnoInicial + intervalo.MesInicial +
           intervalo.Inicial + intervalo.HoraInicial;
14         filaServidor ← obtém NomeServidor da tabela T_Filas onde idFila ==
           intervalo.IdFila ;
15         servidor ← primeiro servidor em filaServidor;
16         while periodo <= intIni do
17             F[servidor] ← F[servidor]U{"nfcapd." + periodo};
18             periodo ← periodo − 5minutos;
19             servidor ← próximo servidor em filaServidor;
20         end
21         intervalo ← intervalo anterior da tabela T_Dados ;
22     end
23 end
```

Inicialmente, o algoritmo de particionamento recebe o período inicial e final obtido da consulta. Após, ele cria uma lista com valores vazios para cada servidor Si da arquitetura (linha 2). Em seguida, o valor do período final (ou período mais recente na linha do tempo) é atribuído a variável *periodo* (linha 3). Com o primeiro servidor identificado pela variável *servidor* (linha 4), o algoritmo inicia um laço de repetição que irá distribuir os arquivos considerados quentes igualmente entre todos os servidores ativos no sistema (linhas 5 à 9).

Ao final desta etapa, a variável *periodo* passa a armazenar informações sobre os arquivos frios. Assim, o algoritmo busca na tabela T_Dados o registro que contém as

informações sobre a localização do conjunto de dados associados à variável *periodo* (linha 10). No passo seguinte, um novo laço de repetição é criado para o tratamento dos dados frios (linhas 11 à 22). Nesta etapa, o algoritmo busca as informações sobre o intervalo de tempo inicial e final no qual a variável *periodo* está contida (linhas 12 e 13). Na sequência, é criada uma lista com os nomes dos servidores que possuem réplicas dos arquivos compreendidos no intervalo do período solicitado (linha 14). No próximo passo do algoritmo, é criado outro laço de repetição (linhas 16 à 20) que irá distribuir os arquivos frios daquele intervalo de tempo de maneira igual entre todos os servidores que possuem as réplicas dos arquivos.

Ao final do processo, todos os servidores recebem a consulta Q com o seu respectivo subconjunto de dados sobre o qual irá realizar o processamento parcial da consulta. Para exemplificar, considere que uma consulta Q é executa sobre o período P que inicia no dia 04 de agosto de 2015 às 00:00 horas até o dia 05 de agosto do mesmo ano às 23:55 horas. Isto significa que os servidores deverão processar um total de 576 arquivos, iniciando com o arquivo chamado `nfcapd.201508040000` e terminando com o arquivo `nfcapd.201508052355`. Considere também a estrutura apresentada anteriormente exemplificado na Figura 5.3, com 4 servidores distribuídos em 2 filas. Ainda, neste exemplo, os arquivos foram distribuídos como sendo todos do dia 05 considerados quentes, e, os arquivos do dia 04 foram divididos entre as filas 01 e 02 utilizando o intervalo de 12 horas.

Ao processar esta consulta exemplo, o algoritmo de particionamento irá criar 4 subconjuntos de dados. Para cada servidor, ele irá dividir os arquivos do dia 05 igualmente entre os 4 servidores, ficando cada um deles com 72 arquivos para serem processados. Para os dados do dia 04, o algoritmo irá dividir os arquivos compreendidos entre às 00:00 horas e 11:55 horas que pertencem a Fila01 entre os servidores A e C. Para os arquivos compreendidos entre 12:00 horas e 23:55 horas, pertencentes à Fila02, o algoritmo irá dividi-los igualmente entre os servidores B e D. Deste modo, cada servidor irá processar um total de 144 arquivos entre quentes e frios.

5.6 Implementação

A implementação do módulo QueryDAS-cliente foi realizada utilizando a linguagem de programação PHP, a mesma da ferramenta NfSen. A integração com a ferramenta ocorre por meio da adição de uma linha em um dos arquivos originais. Após esta modificação, o módulo está pronto para ser utilizado. O módulo QueryDAS-servidor foi implementado também com a linguagem PHP. Nesta versão porém, ele não atua integrado ao NfSen, mas sim como um serviço do nodo servidor que fica aguardando requisições. Para acessar o SAD, os módulos interagem com o sistema de arquivos por meio de sua API na linguagem C.

A implementação do módulo Metadados foi realizada com a utilização de uma biblioteca chamada SQLite [37]. Esta biblioteca é incorporada à linguagem PHP, facilitando assim sua utilização.

Para o armazenamento dos dados de monitoramento foi utilizado o sistema de arquivos distribuído Ceph. Conforme descrito na subseção 4.3.1.3, o Ceph possui o recurso de controle de localidade. Desta forma, pode-se organizar o armazenamento dos dados de acordo com sua utilização, classificando os arquivos como frios e quentes como exemplificado nas seções anteriores. Na atual implementação da arquitetura, a distribuição dos dados é diretamente configurada no sistema de arquivos distribuídos Ceph. Para isso, foi incluído no CRUSH *Map* uma regra chamada $replicaTotal$ que abrange todas as unidades físicas de armazenamento, e em seguida, foi associado um *pool* chamado *dadosQuentes* à esta regra. Desta forma, todos os dados armazenados neste *pool* estão disponíveis em todas as máquinas do ambiente. Contudo, o presente trabalho não considera o acesso exclusivo às réplicas primárias dos arquivos conforme descrito na subseção 4.3.1.3. Assim, eventualmente ocorrem trocas de arquivos entre os servidores que possuem réplicas de um mesmo conjunto de dados.

A atual versão deste trabalho não considera a implementação dos módulos StoreDAS. Assim, as informações armazenadas nos metadados foram preenchidas utilizando instruções SQL via linha de comando. Desta forma, a tabela T_Dados pode ser utilizada para

indicar a localização dos grupos de arquivos em relação aos conjuntos de servidores. Além disso, a inserção dos arquivos contendo os dados de monitoramento no sistema de arquivos distribuído também foi realizada utilizando a interface de linha de comando.

5.7 Sumário

Este capítulo apresentou a arquitetura DASFlow com seus componentes e funcionalidades. O arquivo de configuração é fornecido como valor de entrada para a arquitetura. A partir dele, os módulos StoreDAS-Cliente e StoreDAS-Servidor podem organizar a distribuição dos arquivos armazenados. Para isso, os módulos StoreDAS alimentam o módulo de Metadados com informações sobre os conjuntos de dados e a localização destes no ambiente. Desta forma, o módulo QueryDAS-Cliente pode encontrar os arquivos solicitados em cada consulta, e, particionar o volume de dados a ser processado com os nodos servidores. Embora a arquitetura proposta contemple os três aspectos da escalabilidade, a implementação da solução não está completa e dá suporte apenas à escalabilidade de armazenamento e processamento. Contudo, os experimentos do capítulo 6 mostram o potencial da arquitetura como um todo.

CAPÍTULO 6

EXPERIMENTOS E RESULTADOS

Este capítulo descreve o ambiente e os experimentos utilizados para a validação deste trabalho. Em seguida, apresenta e discute os resultados apontando as vantagens e desvantagens da arquitetura DASFlow.

6.1 Ambiente de Experimentos

A fim de validar a solução da arquitetura DASFlow apresentada no capítulo 5, um ambiente de testes foi elaborado com máquinas servidores do PoP-PR [1]. Nesta arquitetura, foram alocadas máquinas para executar os módulos QueryDAS-Cliente, QueryDAS-Servidor e Metadados. Embora possam atuar na mesma máquina, para este trabalho, os módulos cliente e servidor são executados em máquinas distintas.

Para a etapa de coleta e armazenamento, este trabalho não considera a existência dos módulos de StoreDAS-cliente e StoreDAS-servidor. As funcionalidades oferecidas por estes módulos foram aplicadas de modo manual no ambiente de testes. Isto é, o administrador de redes inseriu, por meio de comandos aplicados diretamente nos servidores, os arquivos no SAD e as informações utilizadas nos metadados.

Para os experimentos, foram utilizadas 5 máquinas. Uma atuando como nodo cliente e quatro desempenhando o papel de nodo servidor. As máquinas disponíveis para a realização dos experimentos possuem as seguintes configurações.

Nodo Cliente: Uma máquina com:

- 2 Processadores Intel Xeon 2.8 GHz com 4 núcleos cada, totalizando 8 cores.
- 32 GB de memória RAM.
- 1 Disco SATA de 7.200 RPM com 750 GB de capacidade.

[1] http://www.pop-pr.rnp.br

- Sistema operacional Ubuntu Server 12.04.

O nodo cliente executa o módulo QueryDAS-Cliente e o módulo de Metadados, isto é, recebe e trata a consulta, envia para os demais nodos, executa o processamento final e devolve o resultado para o NfSen.

Nodo Servidor: Quatro máquinas sendo:

- Todas as máquinas com 2 Processadores Intel Xeon 2.8 GHz com 4 núcleos cada, totalizando 8 cores.
- Todas as máquinas com 32 GB de memória RAM.
- 2 máquinas com 1 Disco cada no padrão SATA de 7.200 RPM com 320 GB de capacidade.
- 2 máquinas com 1 Disco cada no padrão SATA de 7.200 RPM com 750 GB de capacidade.
- Todas as máquinas com Sistema operacional Ubuntu Server 12.04.

As máquinas que atuam como nodo servidor executam o módulo QueryDAS-Servidor, logo, atendem as requisições vindas do nodo cliente. Além disso, elas possuem o Nfdump para processar as consultas parciais.

Para a ferramenta NfSen/Nfdump, o período de tempo selecionado em uma consulta corresponde a quantidade de arquivos que devem ser processados pela ferramenta. Logo, quanto maior for o período, maior será o volume de dados a ser processado. Desta forma, as consultas utilizadas nos experimentos foram classificadas de acordo com o intervalo de tempo solicitado em cada uma delas. Assim, pode-se obter comparações para pequenos, médios e grandes volumes de dados.

Os experimentos consideram dados originários do tráfego do PoP-PR. Foram utilizadas 7 bases de dados, descritas a seguir:

D1: Arquivos obtidos no intervalo de 1 dia (24h), totalizando 18,7 GB de dados.

D2: Arquivos obtidos no intervalo de 1,5 dias (36h), totalizando 28,1 GB de dados.

D3: Arquivos obtidos no intervalo de 2 dias (48h), totalizando 36,5 GB de dados.

D4: Arquivos obtidos no intervalo de 4 dias (96h), totalizando 75 GB de dados.

D5: Arquivos obtidos no período entre 00:05 e 08:30 do mesmo dia, totalizando 3,5 GB de dados.

D6: Arquivos obtidos no período entre 00:05 e 02:00 do mesmo dia, totalizando 800 MB de dados.

D7: Arquivos obtidos no período entre 10:00 e 20:30 do mesmo dia, totalizando 10,9 GB de dados.

As bases de dados D1, D2, D5 e D6 utilizam dados considerados quentes. As bases D3 e D4 utilizam dados quentes e frios e a base D7 utiliza somente dados frios. Os dados quentes e frios são assim classificados em função do período de tempo que cada base de dados representa. Esta diferença entre as bases permite avaliar o funcionamento do módulo de Metadados ao ser requisitado sobre a localização dos dados. A base de dados D7 foi utilizada com o propósito de mostrar o impacto gerado nas consultas em função do intervalo de tempo utilizado para o armazenamento dos dados frios. Para isso, os arquivos desta base estão divididos entre as filas 01 e 02 considerando como divisor as 12:00 horas do dia. Logo, os arquivos armazenados entre 00:00 e 11:55 estão na fila 01 (servidores A e C) e os arquivos armazenados entre 12:00 e 23:55 estão na fila 02 (servidores B e D).

Sobre estes conjuntos de dados, foram executadas dois tipos de consultas: (SF) sem filtro e (CF) com filtro. Para as consultas com filtro foram aplicados dois tipos: **CF-Porta** selecionando apenas o tráfego HTTP (porta 80), e, **CF-Rede** selecionando apenas o tráfego da rede 204.235.231.0/24. A escolha dos valores aplicados no elemento filtro consideram uma síntese das consultas mais utilizadas pelos administradores de rede. Em geral, são utilizadas portas de comunicação para separar o tráfego por tipo, e, endereços de redes e máquinas para identificar origens e destinos nas comunicações. A Tabela 6.1 mostra um resumo das consultas utilizadas no experimentos.

Como pode-se observar na Tabela 6.1, cada linha representa uma consulta utilizada nos experimentos. Cada consulta possui um identificador, o período no qual a consulta é aplicada, o filtro utilizado e o volume de dados a ser processado. Para exemplificar, considere a linha 2 da tabela. Esta consulta é identificada como $D1 - SF$, o período

	A	B	C	D
1	**Consulta**	**Período**	**Filtro aplicado**	**Tamanho da Base**
2	D1-SF	1 Dia	any	18,7 GB
3	D1-CF-Porta		port 80	
4	D2-SF	1,5 Dias	any	28,1 GB
5	D2-CF-Porta		port 80	
6	D2-CF-Rede		net 204.235.231.0/24	
7	D3-SF	2 Dias	any	36,5 GB
8	D3-CF-Porta		port 80	
9	D3-CF-Rede		net 204.235.231.0/24	
10	D4-SF	4 Dias	any	75 GB
11	D4-CF-Porta		port 80	
12	D4-CF-Rede		net 204.235.231.0/24	
13	D5-SF	8 Horas	any	3,5 GB
14	D5-CF-Porta		port 80	
15	D6-SF	2 Horas	any	800 MB
16	D6-CF-Porta		port 80	
17	D7-SF	10,5 Horas	any	10,9 GB
18	D7-CF-Rede		net 204.235.231.0/24	

Tabela 6.1: Resumo das consultas utilizadas nos experimentos.

solicitado é de 1 dia, a consulta não utiliza filtro (*any*) e possui como volume de dados 18,7 GB. De maneira similar, a linha 9 identifica a consulta $D3-CF-Rede$. Esta consulta corresponde ao período de 2 dias, possui filtro da Rede 204.235.231.0/24 e processa um volume de dados de 36,5 GB.

Além do ambiente com 5 máquinas servidores, outro conjunto de máquinas com 2 nodos servidores e 1 nodo cliente foi utilizado para a execução das consultas sobre os conjuntos de dados D5 e D6. Para este ambiente, o nodo cliente possui um processador de 2.6 GHz, 2GB de memória RAM e 500 GB de disco a 7.200 RPM. Os nodos servidores são idênticos em configuração e possuem processador de 3.2 GHz, 4 GB de memória RAM e 300 GB de disco a 7.200 RPM.

6.2 Resultados

Os resultados obtidos nos experimentos são apresentados na Tabela 6.2. Os valores referem-se ao tempo de resposta das consultas em segundos. Para obtê-los foram realizadas 5 consultas, sendo as duas primeiras executadas como aquecimento. A média aritmética das 3 últimas consultas é o tempo de resposta apresentado. Para comparar

os resultados, a mesma consulta é aplicada no NfSen original. O NfSen original é a ferramenta sendo executada na sua formatação clássica, em apenas um nodo. De modo a manter a igualdade nos experimentos, ela foi testada no nodo cliente, o mesmo nodo onde é executado o NfSen da arquitetura DASFlow. A análise dos resultados é descrita a seguir.

	A	B	C	D	E	F	G
	Consulta	**Volume de dados processado (Entrada)**	**Volume de dados processados (Saída)**	**NfSen Original (segundos)**	**Arquitetura DASFlow (segundos)**	**Diferença (segundos)**	**Percentual (%)**
1	D1-SF	18,7 GB	18,7 GB	586	689	103	17,6
2	D1-CF-Porta		10,3 GB	166	133	-33	-19,9
3	D2-SF	28,1 GB	28,1 GB	1182	1384	202	17,1
4	D2-CF-Porta		15,1 GB	299	203	-96	-32,1
5	D2-CF-Rede		1,9 GB	297	194	-103	-34,7
6	D3-SF	36,5 GB	36,5 GB	4921	5167	246	5,0
7	D3-CF-Porta		23,5 GB	1039	895	-144	-13,9
8	D3-CF-Rede		2,7 GB	370	243	-127	-34,3
9	D4-SF	75 GB	75 GB	10376	10660	284	2,7
10	D4-CF-Porta		44 GB	1774	1669	-105	-5,9
11	D4-CF-Rede		5,6 GB	1244	1022	-222	-17,8
12	D5-SF	3,5 GB	5,3 GB	114,5	269,7	155,2	135,5
13	D5-CF-Porta		**1,4 GB**	**69,2**	**59,1**	**-10,1**	-14,6
14	D6-SF	800 MB	2,6 GB	21,3	58,3	37	173,7
15	D6-CF-Porta		**300 MB**	**12,4**	**10,2**	**-2,2**	-17,7
16	D7-SF	10,9 GB	10,9 GB	658	775	117	17,8
17	D7-CF-Rede		770 MB	84	84	0	0,0

Tabela 6.2: Tempo de resposta (em segundos) das consultas executadas.

Ao apresentar os resultados, nota-se que as consultas do tipo com filtro (CF) executadas na arquitetura DASFlow apresentaram melhor desempenho se comparadas ao tempo de resposta obtido pela ferramenta NfSen original. Para este tipo de consulta, o resultado parcial possui um volume de dados consideravelmente menor, de acordo com o filtro aplicado. Observa-se que nas consultas D2-CF-Rede, D3-CF-Rede e D4-CF-Rede o tamanho total dos arquivos enviados pelos nodos servidores para o nodo cliente corresponde aproximadamente a menos de 10% do tamanho dos dados de entrada. Deste modo, o tempo de resposta das consultas, incluindo o tempo de envio e processamento dos resultados parciais dos nodos servidores para o nodo cliente somando ao processamento final, possui um ganho que varia entre 13% e 34% (células G5, G8 e G11).

Para as consultas que utilizam o parâmetro porta de comunicação no filtro (D1-CF-Porta, D2-CF-Porta, D3-CF-Porta, D4-CF-Porta, D5-CF-Porta e D6-CF-Porta), nota-se

que a diferença no tempo de resposta não é tão expressivo quanto as que aplicam filtro de rede. Isto ocorre porque o resultado parcial dos nodos servidores possuem um conjunto maior de dados, logo, o volume gerados na saída varia entre 30% e 60% em relação ao volume de entrada. Isto é, o ganho no desempenho é prejudicado pelo maior tempo de envio e processamento dos resultados parciais dos nodos servidores para o nodo cliente.

É importante salientar que as consultas mais utilizadas em cenários reais pelos administradores de rede são as consultas com filtro e sobre dados recentes. Isto ocorre porque os dados de monitoramento tendem a ser menos acessados à medida que se tornam antigos. Além disso, a utilização de filtros auxilia o administrador a eliminar resultados menos importantes para determinadas consultas. Por exemplo, considere que houve um aumento no tráfego de rede com início às 00:30 horas até as 6:30 horas do mesmo dia. Este aumento se comparado com outros dias representa uma anomalidade no comportamento do tráfego. A partir disto, o administrador submete uma consulta sem filtro para obter uma listagem de quais origens e destinos são responsáveis pela maior parte do tráfego correspondente ao período. Com isso, o administrador dispara uma série de consultas utilizando como filtros os resultados da consulta anterior. O objetivo das consultas com filtro é investigar os relacionamentos entre as origens e destinos, as portas de comunicação utilizadas, a quantidade de *bytes* trafegados entre elas, e assim por diante. Desta forma, pode-se identificar o mais breve possível os causadores do comportamento anômalo a fim de executar um plano de ação para corrigir o problema, ou, apenas efetuar registros sobre o tráfego caso este seja legítimo.

Para as consultas do tipo sem filtro (SF) quando executadas na arquitetura DASFlow possuem tempo de resposta maior em relação à ferramenta NfSen em seu formato original. Pode-se observar que a consulta D6-SF chega a ser aproximadamente 173% mais lenta enquanto que a consulta D5-SF é 135% mais lenta em relação a consulta na ferramenta original. Para estas consultas, observa-se que o volume de dados gerado como saída pelos nodos servidores possui tamanho aproximadamente 3 vezes maior que a entrada (células C12 e C14). Nestas consultas, os arquivos temporários que armazenam os resultados parciais não possuem compressão de dados. Desta forma, há um maior impacto no tempo

de envio e processamento destes arquivos pelos nodos servidores para o nodo cliente.

Para as demais consultas (D1-SF, D2-SF, D3-SF, D4-SF e D7-SF) foi aplicada a compressão de dados nos arquivos temporários, isto significa que o volume de dados na saída é equivalente ao volume processado na entrada. Embora a diferença entre as consultas D5-SF e D6-SF para as demais sem filtro (SF) tenha diminuído por conta da compressão, o desempenho continua maior em relação à ferramenta original. Isso ocorre porque há um custo no acesso aos arquivos armazenados no DFS Ceph associados ao tempo de transmissão dos arquivos temporários criados pelos nodos servidores e enviados ao nodo cliente. Isto significa um maior tempo de resposta de até 17,8% em relação a consulta na ferramenta original (célula G16).

O conjunto de dados D7 foi utilizado nos experimentos para demonstrar como o armazenamento dos arquivos com dados frios pode interferir no desempenho de uma consulta. Como estes arquivos não possuem réplica total, uma consulta pode sofrer atraso no tempo de resposta pela sobrecarga de alguns nodos servidores em relação aos demais. Nota-se que a consulta com filtro (D7-CF-Rede) deveria obter melhor desempenho se comparada à ferramenta original. O processamento desta consulta no entanto, exige que os nodos servidores B e D processem o volume de dados do período entre 12:00 e 20:30 enquanto que os nodos A e C processam os arquivos do período entre 10:00 e 11:55, baseado no intervalo de tempo utilizado para o agrupamento dos dados armazenados conforme descrito anteriormente. Desta forma, a consulta tem um acréscimo no tempo de resposta devido ao maior tempo demandado pelos servidores B e D para processarem um maior volume de dados em relação aos servidores A e C.

Os resultados apresentados demonstram que o ganho no tempo de resposta das consultas aplicadas na arquitetura DASFlow está relacionado ao filtro aplicado. Quanto mais resultados são filtrados nos nodos servidores, menores são os tamanhos dos resultados parciais, ou seja, menor é o tempo de envio gasto pelos nodos servidores e menor é o tempo de processamento gasto pelo nodo cliente. Adicionalmente, a organização do armazenamento dos arquivos com dados frios deve ser planejada a fim de evitar que as consultas sobre estes dados (consultas gerenciais), sejam prejudicadas por conta da sobrecarga no

processamento de alguns nodos servidores em relação aos demais.

6.3 Sumário

Como pode-se observar neste capítulo, a arquitetura proposta para prover escalabilidade de processamento e armazenamento para a ferramenta de monitoramento NfSen/Nfdump atende parcialmente estes requisitos. Embora apenas as consultas com filtro obtiveram melhor tempo de resposta, elas são utilizadas em maior número e com mais frequência se comparadas as consultas sem filtro. As consultas mais utilizadas buscam identificar, por exemplo, quem é responsável pelo tráfego de rede (filtro de origens e destinos) e que tipo de comunicação é estabelecida entre as origens e destinos (filtro de portas de comunicação). Assim, os administradores de rede podem executar planos de ação baseados nos resultados das consultas. Para o armazenamento, a utilização de um sistema de arquivos distribuído possibilita o aumento dinâmico da capacidade de armazenamento da ferramenta. Isto é, pode-se adicionar servidores ao ambiente à medida que aumenta a demanda por espaço de armazenamento. Para o processamento, a arquitetura mostrou-se eficiente para as consultas que utilizam filtros. Porém, para as consultas sem filtro a ferramenta NfSen/Nfdump em seu formato original apresentou melhores resultados. Assim, a arquitetura DASFlow deve ser aprimorada para melhorar os tempos de respostas para todos os tipos de consultas.

CAPÍTULO 7

CONCLUSÃO E TRABALHOS FUTUROS

Este capítulo descreve as conclusões obtidas relacionando a proposta deste trabalho frente aos resultados alcançados nos experimentos. Adicionalmente, discute algumas questões pendentes e que devem ser abordadas nos trabalhos futuros.

7.1 Conclusão

Neste trabalho foi apresentada uma arquitetura distribuída para fornecer escalabilidade de processamento, armazenamento e coleta para dados de monitoramento de rede. Esta arquitetura é projetada e aplicada para a ferramenta NfSen/Nfdump. Ela é concretizada pela implementação dos módulos de Metadados e QueryDAS. Os módulos integram nodos remotos extraindo seus recursos de processamento e armazenamento fazendo-os trabalhar de forma cooperativa. A avaliação da solução traz alguns aspectos relevantes descritos na sequência.

O primeiro aspecto está relacionado as consultas do tipo com filtro (CF). Os resultados obtidos nos experimentos mostraram a viabilidade imediata da solução para o processamento de consultas com caráter investigativo. Isto é, aquelas que buscam informações relacionadas a incidentes de segurança e que são as mais utilizadas e demandam menor tempo de resposta. Ainda que o ganho no tempo de resposta esteja vinculado ao filtro aplicado, a arquitetura traz como vantagem a existência de um ambiente distribuído que provê poder de processamento aliado ao armazenamento escalável dos dados de monitoramento.

O outro aspecto está relacionado as consultas do tipo sem filtro (SF). Quando executadas na ferramenta original, estas consultas possuem melhor desempenho frente as consultas sem filtro aplicadas na arquitetura DASFlow. Isso demonstra que a abordagem utilizada pela arquitetura, de interferir o mínimo possível na ferramenta original, não se

mostrou uma boa alternativa para este caso. Ainda que o aspecto de armazenamento seja aplicável de forma escalável pela arquitetura, o aspecto do processamento distribuído precisa ser melhorado para que as consultas sem filtro (SF) possam ser utilizadas obtendo menor tempo de resposta.

De modo complementar, este trabalho considerou apenas a implementação dos módulos QueryDAS-Cliente, QueryDAS-Servidor e Metadados. Isto significa que a escalabilidade proposta pela solução tratou apenas os aspectos de processamento e armazenamento. Com isso, a escalabilidade de coleta, fornecida pelos módulos StoreDAS, é uma das limitações que devem ser consideradas nos trabalhos futuros.

Os resultados parciais deste trabalho, relacionados às consultas com os conjuntos de dados D5 e D6, foram publicados no artigo *DASFlow: Uma Arquitetura Distribuída de Processamento e Armazenamento para Dados de Monitoramento de Rede*, no XX Workshop de Gerência e Operação de Redes e Serviços (WGRS2015).

7.2 Trabalhos Futuros

Para os trabalhos futuros, são planejados estudos, implementações e experimentos com abordagens alternativas para o processamento das consultas distribuídas. Com isso, espera-se que tanto consultas sem filtro (SF) quanto consultas com filtro (CF) possam obter melhor desempenho na arquitetura DASFlow em relação a ferramenta original, indiferentemente do filtro aplicado ou do volume de dados a ser processado.

Para isso, considera-se a utilização de uma saída alternativa como resultado parcial dos nodos servidores. Esta saída deve conter como resultado da consulta o conteúdo que será impresso na tela do administrador. Com isso, o volume de informações trocadas entre os nodos cliente e servidores possui pouca variação entre consultas com e sem filtro. Uma alternativa para isso seria a utilização de arquivos com os resultados em forma de tabelas armazenados em um arquivo com o formato "csv". Assim, o nodo cliente realizaria apenas a formatação visual do conteúdo apresentando o resultado final para o administrador. Esta solução contudo, deve ser estudada e planejada para que o resultado seja idêntico ao da ferramenta original.

Para a etapa de coleta, é planejada uma interface adicional no módulo Metadados para atuar em conjunto com os módulos StoreDAS-Cliente e StoreDAS-Servidor. Com isso, será possível realizar o agrupamento dinâmico dos dados coletados de acordo com a importância temporal. À medida em que novos dados são coletados, os antigos serão movidos e armazenados propositadamente somente em alguns nodos, evitando réplica total e garantindo o aumento na capacidade de armazenamento. Este recurso será provido pela implementação dos módulos StoreDAS-Cliente e StoreDAS-Servidor atuando em conjunto com o módulo de Metadados.

Adicionalmente, esta solução pode ser utilizada para integrar diversas redes permitindo que o NfSen tenha acesso a um conjunto maior de dados de monitoramento. Para isto, os módulos cliente e servidor somados ao metadados, atuarão para integrar a coleta e armazenamento de todos os nodos de tal forma que os fluxos IP estarão disponíveis para qualquer consulta, ou seja, sobre dados recentes ou antigos. Desta forma, a partir de uma única consulta da ferramenta NfSen poderá ser obtida uma visão geral de todas as redes monitoradas, não sendo necessárias várias consultas para contemplar todos os dados coletados.

APÊNDICE A

APÊNDICE

O Apêndice A apresenta a Tabela A.1 com os parâmetros e valores de cada elemento utilizado nas consultas da ferramenta NfSen/Nfdump.

Elemento	Parâmetro	Valor			
[Origem]	- M	/caminho/para/primeiro-diretorio:proximo-diretorio:ultimo-diretorio			
[Período]	- r	Arquivo simples (Ex. nfcapd.201508041930)			
	- R	Múltimos arquivos (Ex. nfcapd.201508041930:nfcapd.201508051900)			
[Filtro]	expr	Os parâmetros 'expr' podem assumir os seguintes valores:			
	Qualquer valor	any			
	Versão do protocolo	inet, ipv4, inet6, ipv6			
	Protocolo	TCP, UDP, ICMP, GRE, ESP, AH, RSVP			
	Endereço IP	IP a.b.c.d ou HOST a.b.c.d			
	Origem	SRC + <Endereço IP>			
	Destino	DST + <Endereço IP>			
	Rede	NET a.b.c.d/<num-mask> ou a.b.c.d m.n.r.s			
	Porta	PORT [comp] num-port			
	Interface	IF + num-if			
	IN/OUT	IF + num-if IN/OUT			
	Flags	TCP Flags: (A)ACK/ (S)SYN/ (F)FIN/ (R)RESET/ (P)PUSH/ (U)URGENT/ (X) ALL			
	ToS (Tipo de Serviço)	ToS-num 0-255			
	Pacotes	packets [comp] num [scale]			
	Bytes	bytes [comp] num [scale]			
	Packets por segundo	pps [comp] num [scale]			
	Bits por segundo	bps [comp] num [scale]			
	Duração (milisegundos)	duration [comp] num			
	Bytes por pacotes	bpp [comp] num [scale]			
	AS (Autonomous System)	AS-num			
	Scale	K, M, G			
	Comp	=, ==, >, <, EQ, LT, GT (= é assumido como padrão)			
[Opções]					
Limite	- c	<num>			
Agregação	- A	SRCIP, DSTIP, SRCPORT, DSTPORT			
Saída	- o fmt: <tag>	<tag>:			
		Tag	Descrição	Tag	Descrição
		%ts	Tempo Inicial	%in	Interface de Entrada
		%te	Tempo Final	%out	Interface de Saída
		%td	Duração	%pkt	Pacotes
		%pr	Protocolo	%byt	Bytes
		%sa	Endereço Origem	%fl	Fluxos
		%da	Endereço Destino	%flg	TCP Flags
		%sap	Endereço Origem:Porta	%tos	Tos
		%dap	Endereço Destino:Porta	%bps	bps - bits por segundo
		%sp	Porta Origem	%pps	pps - pacotes por segundo
		%dp	Porta Destino	%bpp	bps - Bytes por pacote
		%sas	AS Origem		
		%das	AS Destino		
Estatísticas	- s <stat>/<order>	**<stat>**			
		record	Estatística sobre registros NetFlow agregados		
		srcip	Estatística sobre endereços IP de origem		
		dstip	Estatística sobre endereços IP de destino		
		ip	Estatística sobre quaisquer (origem ou destino) endereços IP		
		srcport	Estatística sobre portas de origem		
		dstport	Estatística sobre os portas de destino		
		port	Estatísticas sobre qualquer (origem ou destino) portas		
		srcas	Estatística sobre fonte do AS (números)		
		dstas	Estatística sobre destino do AS (números)		
		as	Estatística sobre qualquer (origem ou destino) de AS (números)		
		inif	Estatística sobre números de interface de entrada		
		outif	Estatística sobre números de interface de saída		
		if	Estatística sobre quaisquer (entrada ou saída) números de interface		
		proto	Estatística sobre números de protocolo		
		<order>			
		flows	Fluxos		
		packets	Pacotes		
		bytes	Bytes		
		pps	bps - bits por segundo		
		bps	pps - pacotes por segundo		
		bpp	bps - Bytes por pacote		
Top N Estatísticas	- n <num> - s <stat>/<order>	<num> = 0 - N			

Tabela A.1: Parâmetros e valores de cada elemento utilizado nas consultas NfSen/Nfdump.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] API. Api - application programming interface. `http://www.npr.org/api/index`. Acessado em julho de 2015.

[2] Nevil Brownlee, Cyndi Mills, e Greg Ruth. Traffic flow measurement: Architecture. `http://tools.ietf.org/html/rfc2722`, 1999. Acessado em junho de 2014.

[3] B. Claise. Cisco systems netflow services export version 9. rfc 3954 (informational), outubro de 2004.

[4] B. Claise. Ipfix protocol. `http://tools.ietf.org/pdf/rfc7011.pdf`, setembro de 2013.

[5] P. Meyer D. Levi. Simple network management protocol (snmp) applications. `https://tools.ietf.org/html/rfc3413`, 2002. Acessado em julho de 2015.

[6] Benjamin Depardon, Gaël Le Mahec, e Cyril Séguin. Analysis of Six Distributed File Systems. Rapport de recherche, SysFera , Laboratoire Modélisation, Information et Systèmes - MIS, Feb de 2013.

[7] Luca Deri. nprobe software. `http://www.ntop.org/products/nprobe/`. Acessado em junho de 2014.

[8] Luca Deri. Ntop software. `http://www.ntop.org/products/ntop/`. Acessado em março de 2014.

[9] Luca Deri, Alfredo Cardigliano, e Francesco Fusco. 10 gbit line rate packet-to-disk using n2disk. *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*. IEEE, 2013.

[10] Luca Deri e Francesco Fusco. Microcloud-based network traffic monitoring. *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013.

[11] Câmara Federal. Marco civil da internet. projeto de lei 2.126/2011. http://www.camara.gov.br/proposicoesWeb/fichadetramitacao?idProposicao=517255. Acessado em abril de 2014.

[12] Mark Fullmer. Flow-tools software. http://www.splintered.net/sw/flow-tools/docs/flow-tools.html. Acessado em março de 2014.

[13] Lei Gao, Jiahai Yang, Hui Zhang, Bin Zhang, e Donghong Qin. Flowinfra: a fault-resilient scalable infrastructure for network-wide flow level measurement. *Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific*. IEEE, 2011.

[14] GlusterFS. Gluster file system. http://www.gluster.org. Acessado em maio de 2014.

[15] GlusterFS. Gluster: An introduction to gluster architecture, 2011.

[16] The OpenNMS Group. Opennms. http://www.opennms.org. Acessado em julho de 2015.

[17] Hadoop. The hadoop distributed file system. http://aosabook.org/en/hdfs.html. Acessado em maio de 2014.

[18] HDFS. Hadoop distributed file system. http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. Acessado em maio de 2014.

[19] iRODS. Data grids, digital libraries, persistent archives, and real-time data systems. https://wiki.irods.org/index.php/IRODS:Data_Grids,_Digital_Libraries,_Persistent_Archives,_and_Real-time_Data_Systems. Acessado em maio de 2014.

[20] iRODS. Integrated rule-oriented data system. http://irods.org. Acessado em maio de 2014.

[21] Yeonhee Lee e Youngseok Lee. Toward scalable internet traffic measurement and analysis with hadoop. *ACM SIGCOMM Computer Communication Review*, 43(1), 2013.

[22] Youngseok Lee, Wonchul Kang, e Hyeongu Son. An internet traffic analysis method with mapreduce. *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*. IEEE, 2010.

[23] Eliezer Levy e Abraham Silberschatz. Distributed file systems: Concepts and examples. *ACM Computing Surveys (CSUR)*, 22(4), 1990.

[24] Jun Li, Shuai Ding, Ming Xu, Fuye Han, Xin Guan, e Zhen Chen. Tifa: Enabling real-time querying and storage of massive stream data. *Networking and Distributed Computing (ICNDC), 2011 Second International Conference on*. IEEE, 2011.

[25] Zabbix LLC. Zabbix. `http://www.zabbix.com`. Acessado em julho de 2015.

[26] Cristian Morariu, Thierry Kramis, e Burkhard Stiller. Dipstorage: Distributed architecture for storage of ip flow records. *Proc. of the 16th Workshop on Local and Metropolitan Area Networks*, 2008.

[27] Cristian Morariu, Peter Racz, e Burkhard Stiller. Script: a framework for scalable real-time ip flow record analysis. *Network Operations and Management Symposium (NOMS), 2010 IEEE*. IEEE, 2010.

[28] Cristian Morariu e Burkhard Stiller. Dicap: Distributed packet capturing architecture for high-speed network links. *Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on*. IEEE, 2008.

[29] Nagios. Nagios. `https://www.nagios.org`. Acessado em julho de 2015.

[30] NfSen/Nfdump. `http://nfsen.sourceforge.net`. Acessado em março de 2014.

[31] Official Journal of the European Union. Directive 2006/24/ec of the european parliament and of the council, março de 2006.

[32] Cisco White Paper. Cisco visual networking index: Forecast and methodology, 2012-2017, maio de 2013.

[33] Siamak Sarmady. A survey on peer-to-peer and dht. *arXiv preprint arXiv:1006.4708*, 2010.

[34] sFlow Protocol. `http://sflow.org/sflow_version_5.txt`, maio de 2004.

[35] Abraham Silberschatz, Henry F Korth, e S Sudarshan. *Database System Concepts, Fourth Edition*, volume 1. McGraw-Hill New York, 2001.

[36] Caida Software. Flow-scan software. `http://www.caida.org/tools/utilities/flowscan/`. Acessado em abril de 2014.

[37] SQLite. `http://www.sqlite.org`. Acessado em agosto de 2015.

[38] Andrew TANENBAUM. S. redes de computadores. são paulo: Ed, 2003.

[39] TCPDUMP/LIBPCAP. `http://www.tcpdump.org`. Acessado em março de 2014.

[40] Tran Doan Thanh, Subaji Mohan, Eunmi Choi, SangBum Kim, e Pilsung Kim. A taxonomy and survey on distributed file systems. *Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on*, volume 1. IEEE, 2008.

[41] Sage A Weil. Ceph. `http://ceph.com/docs/master/`. Acessado em março de 2014.

[42] Sage A Weil, Scott A Brandt, Ethan L Miller, e Carlos Maltzahn. Crush: Controlled, scalable, decentralized placement of replicated data. *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*. ACM, 2006.

[43] Sage A Weil, Andrew W Leung, Scott A Brandt, e Carlos Maltzahn. Rados: a scalable, reliable storage service for petabyte-scale storage clusters. *Proceedings of the 2nd international workshop on Petascale data storage: held in conjunction with Supercomputing'07*. ACM, 2007.

[44] Wireshark. http://www.wireshark.org/docs/. Acessado em março de 2014.