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

Uma Arquitetura de Big Data as a Service Baseada no Modelo de Nuvem Privada

Marco Antônio de Sousa Reis

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

Orientador
Profa. Dra. Aletéia Patrícia Favacho de Araújo

Coorientador
Profa. Dra. Maristela Terto de Holanda

Brasília
2018

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

RM321a

Reis, Marco Antonio de Sousa
Uma Arquitetura de Big Data as a Service Baseada no Modelo de Nuvem Privada / Marco Antonio de Sousa Reis; orientador Aletéia Patrícia Favacho de Araújo; co orientador Maristela Terto de Holanda. -- Brasília, 2018.
105 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2018.

1. Big Data. 2. Computação em Nuvem. 3. Nuvem Privada. 4. OpenStack. 5. Big Data as a Service. I. Araújo, Aletéia Patrícia Favacho de , orient. II. Holanda, Maristela Terto de , co-orient. III. Título.

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

Uma Arquitetura de Big Data as a Service Baseada no Modelo de Nuvem Privada

Marco Antônio de Sousa Reis

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

Profa. Dra. Aletéia Patrícia Favacho de Araújo (Orientador)
CIC/UnB

Prof. Dr. Vinod Rebello
IC/UFF

Prof. Dr. Jacir Luiz Bordim
CIC/UNB

Profa. Dra. Aletéia Patrícia Favacho de Araújo
Coordenador do Programa de Pós-graduação em Computação Aplicada

Brasília, 18 de Julho de 2018

Dedicatória

Este trabalho é dedicado à minha família, que compartilhou do esforço e compreendeu as (muitas) horas de dedicação. Que seja uma gota de motivação em suas vidas para que se dediquem à educação. E dedico também à ciência. É um privilégio viver em um Mundo com tantas mentes brilhantes.

Agradecimentos

Neste documento apresento o resultado de um estudo para o Mestrado Profissional em Computação Aplicada na Universidade de Brasília, que foi uma fantástica jornada no mundo acadêmico. Assim, esta dissertação pretende ser uma pequena contribuição para o Brasil. Durante o período de dois anos foram conduzidas pesquisas e experimentos exaustivos, que consumiram muito tempo e energia. Tamanho esforço não seria possível sem a ajuda de muitas pessoas e organizações. Todos merecem meu respeito e agradecimento.

Meu primeiro agradecimento vai para o Brasil, por oferecer um curso de tamanha qualidade. Agradeço também à UnB, sem ela nada disso seria possível. Naturalmente, minhas professoras merecem um agradecimento especial, por tanta dedicação. Por fim, mas igualmente importante, agradeço ao TJDFT, que contribuiu sobremaneira para a conclusão deste trabalho.

Resumo

Este trabalho apresenta uma proposta de arquitetura para plataforma de *big data* como serviço baseada em nuvem privada (Big Data as a Service). Desta forma, são apresentadas as funcionalidades, as técnicas e as ferramentas que compõem o estado da arte nesta linha recente de pesquisa, resultado da união da computação em nuvem com *big data*. Como benefício direto, procura-se encurtar o tempo necessário para adoção efetiva de soluções de *big data* nas organizações. Assim, com a nuvem privada, a arquitetura aproveita os recursos computacionais disponíveis no *datacenter* para disponibilizar as características de computação em nuvem já populares nos principais provedores. Dessa forma, o estudo mostra uma alternativa para minimizar o problema da complexidade da implantação, gerenciamento e configuração do ambiente de nuvem privada e de *big data*, oferecendo aos usuários a possibilidade de usarem as soluções de *big data* com o provisionamento direto dos serviços de análise e de integração de dados, que serão implementados por meio de tecnologias para processamento em *batch*, *real time* e NoSQL. Os resultados mostram que a arquitetura atende aos principais casos de uso de sistemas de *big data*, garantindo características como escalabilidade, modularidade, reusabilidade, uso de padrões abertos etc.

Palavras-chave: Big Data, Computação em Nuvem, Nuvem Privada, OpenStack, Big Data as a Service

Abstract

This work presents a proposal of an architecture for Big Data as a Service (BDaaS). Hence, are explained the features, techniques and tools that compose the state of the art in this recent research line, the result of the union of cloud computing and big data. As a direct benefit, it is sought to shorten the time necessary for effective adoption of big data solutions in organizations. Thus, with the private cloud, the architecture takes advantage of computational resources available in the datacenter to provide the characteristics of cloud computing already popular with major providers. In this way, the study shows an alternative to minimize the problem of deployment, management and configuration complexity in the private cloud environment and big data, allowing the users to use big data solutions with analysis services and data integration services, which will be implemented by means of technologies for batch, real time and NoSQL processing. The results show that the architecture addresses the main use cases of big data systems, enabling features such as scalability, modularity, reusability, use of open standards etc.

Keywords: Big Data, Cloud Computing, Private Cloud, OpenStack, Big Data as a Service

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A quantidade de aplicações disponíveis na Internet aumenta rapidamente, acompanhando a demanda por tecnologia da vida moderna. O Governo Federal e, por conseguinte, o Poder Judiciário da União, seguem essa tendência e oferecem cada vez mais serviços digitais para atender à sociedade por meio de iniciativas como a Lei de Acesso à Informação (LAI) [60], o Modelo Nacional de Interoperabilidade do Poder Judiciário e do Ministério Público [35, 37] e a Política Nacional de Atenção Prioritária ao Primeiro Grau de Jurisdição [38].

Todavia, esse cenário é, ao mesmo tempo, fonte de oportunidades e de desafios. Se por um lado o cidadão tem acesso à facilidade dos serviços disponíveis na Internet, por outro lado o provisionamento desses recursos computacionais representa uma disputa entre o tempo e o custo. Além disso, novos sistemas de informação são desenvolvidos frequentemente, demandando a instalação de novos servidores, que precisam de manutenção e de políticas de *backup*. Consequentemente, a infraestrutura necessária para suportar essa demanda é complexa, tem alto custo financeiro e demanda mão de obra especializada, pois o ambiente de grandes empresas costuma ser heterogêneo, com múltiplos sistemas operacionais, dispositivos de armazenamento, linguagens de programação e servidores de aplicação. Neste contexto, a computação em nuvem tem o objetivo de agilizar e otimizar o uso dos equipamentos no *datacenter* por meio da virtualização, permitindo a melhor utilização e diminuindo a ociosidade dos recursos computacionais [90].

O crescimento da informatização no Governo Federal traz consigo o aumento do volume de dados gerados pelos dispositivos eletrônicos, incluindo computadores, *smartphones*, *tablets*, integrações com outros órgãos e mídias sociais. Considerando que essas informações precisam ser armazenadas e analisadas, essa situação se tornou um desafio para as ferramentas tradicionais de tecnologia. Desta maneira, o governo está constantemente em busca de novos modelos e ferramentas para dar maior celeridade e publicidade aos seus serviços, sendo que suas linhas de atuação compreendem o provimento de infraestrutura e de tecnologia apropriadas ao funcionamento dos seus serviços, o favorecimento da des-

centralização administrativa, bem como o diálogo com a sociedade, incluindo instituições públicas e privadas [38].

Assim sendo, o processamento e a disponibilização desse grande volume de informações são tarefas complexas que exigem um alto poder computacional, o qual frequentemente é superdimensionado para atender a eventuais picos de acesso. Isso impede uma boa relação entre custo e benefício no *datacenter*, uma vez que pode haver ociosidade ou superalocação desnecessária de recursos [20, 130].

A situação na qual há muitos dados a serem processados tem sido chamada de *big data*, um conceito que está relacionado à geração ou ao consumo de um grande volume de dados em curto espaço de tempo, de forma que a infraestrutura tradicional de tecnologia não consegue fazer o processamento eficientemente [27, 41]. Não há uma definição formal da quantidade de dados que pode ser classificada como *big data*, entretanto, nesta dissertação será utilizada a escala de milhões e bilhões de registros.

Atualmente, o baixo custo do armazenamento, as redes de alta velocidade e a evolução dos processadores permitiram a emergência de ferramentas como o Apache Hadoop [6], que é uma reconhecida ferramenta para soluções *big data*. O Hadoop é, então, uma alternativa às ferramentas tradicionais, como *storages* e Sistemas Gerenciadores de Bancos de Dados Relacionais (SGBDR), para o processamento de *big data* [125].

Diante disso, tem-se percebido que a combinação entre essas duas tendências, *big data* e computação em nuvem, é motivo de interesse na indústria de software e no meio acadêmico, levando à criação de uma categoria de tecnologia chamada de *Big Data as a Service* (BDaaS). Todavia, a implantação de soluções BDaaS não é trivial, conforme será conferido no desenvolvimento deste estudo, pois as organizações têm dificuldade para entregar soluções efetivas devido à falta de *expertise* [23, 131].

Diante do exposto, este projeto de pesquisa propõe um modelo de arquitetura para infraestrutura de *datacenter* baseado em nuvem privada para entregar soluções *big data* na forma de serviço. O resultado pretende beneficiar tanto o cidadão quanto o Governo Federal com a entrega de serviços de banco de dados para grandes volumes na Internet, de maneira transparente e eficiente por meio de BDaaS.

Assim, considerando a complexidade do assunto e o objetivo de orientar a implementação, é proposto um *framework* conceitual para classificar as diferentes cargas de trabalho e associá-las com suas respectivas funcionalidades no BDaaS. Com os dados devidamente particionados, a arquitetura proposta provisiona os métodos e a infraestrutura computacional para a transformação dos dados de entrada em sistemas de *big data*.

1.1 Contribuições

Durante os estudos para esta dissertação foram executados diversos experimentos que, juntamente com outros autores, resultaram na publicação de seis artigos em eventos nacionais e internacionais, bem como um capítulo de livro. Estes trabalhos abordam as áreas da computação em nuvem, *big data*, arquitetura de software e geoprocessamento. As publicações, listadas em ordem cronológica, são [40] (concorreu ao prêmio de melhor artigo do evento), [75], [51], [73], [72], [71] e [74].

1.2 Objetivos

O objetivo geral deste trabalho é desenvolver uma arquitetura para sistemas de *big data* como serviço baseada em nuvem privada, detalhando as funcionalidades, as técnicas e as ferramentas mais adequadas para este tipo de plataforma, resultado da união da computação em nuvem com *big data*.

Para alcançar o objetivo geral deste trabalho, faz-se necessário atingir os seguintes objetivos específicos:

- Criar um modelo para relacionar as funcionalidades do *datacenter* tradicional com as ferramentas correspondentes de *big data*;
- Implementar uma ferramenta para realizar tarefas de ETL (*extract, transform, load*) de *big data*, com o objetivo de separar cargas de trabalho de acordo com o tipo de acesso, processamento e padrão de carregamento dos dados;
- Selecionar casos de uso com *datasets* governamentais que serão implementados para validar a proposta;
- Avaliar técnicas de alocação sob demanda e dimensionamento dinâmico de *big data*;
- Avaliar técnicas de armazenamento e gerenciamento de dados;
- Analisar e comparar a performance na infraestrutura proposta com a infraestrutura tradicional;
- Definir um roteiro de implantação do modelo de infraestrutura proposto nas instituições públicas e privadas.

De uma forma geral, a dissertação desenvolve um roteiro sistemático para a implantação e a compreensão destas importantes e emergentes áreas da tecnologia. Com isso, pretende-se acelerar a adoção de sistemas de *big data* em nuvem privada para resolver problemas práticos vivenciados pelas organizações públicas e privadas.

1.3 Metodologia

Esta seção descreve a metodologia utilizada nesta dissertação, assim, serão apresentadas as abordagens, os objetivos e os procedimentos da pesquisa, descritos segundo Gerhardt e Silveira [57]. A arquitetura proposta desenvolve um procedimento sistemático para indicar soluções para problemas identificados em uma determinada realidade, nesse caso, a criação de sistemas de *big data* na nuvem privada.

O primeiro passo é classificar o estudo com relação à abordagem [57]. Neste sentido, esta pesquisa é qualitativa e quantitativa: (i) qualitativa porque descreve as funcionalidades e propostas de soluções, adicionando novos casos de uso dos sistemas de *big data* sem preocupação inicial com a representatividade numérica das medidas de performance; e (ii) quantitativa porque compara, mesmo que parcialmente, a performance da arquitetura proposta com a do *datacenter* tradicional. Essa dificuldade de comparação se deve ao fato de que não há uma correspondência exata entre os serviços e os equipamentos oferecidos pelo *big data* e os serviços do *datacenter* tradicional.

Com relação aos objetivos, esta pesquisa é exploratória e explicativa: (i) exploratória porque investiga e identifica as dificuldade para criar e implantar sistemas de *big data*; e (ii) explicativa porque demonstra as possíveis soluções e formaliza esse conhecimento em uma descrição arquitetural, que pode ser reutilizada em outras organizações [66, 113].

Quanto aos procedimentos, esta é uma pesquisa experimental e de estudo de caso: (i) experimental pois foram avaliadas diversas técnicas e ferramentas, sendo selecionadas as mais adequadas para cada serviço oferecido pelo BDaaS; e (ii) estudo de caso, caracterizado pela implementação da prova de conceito com dados reais para validar cada item da proposta.

Por fim, no que diz respeito à natureza do estudo, esta é uma pesquisa aplicada, por estar orientada à solução de problemas específicos durante a implantação de soluções de *big data* em nuvem privada, uma situação que se repete na indústria e na academia.

1.4 Estrutura do Trabalho

Este trabalho está dividido em 6 capítulos, contando com esta introdução. O Capítulo 2, Referencial Teórico, mostra as definições para *big data* e computação em nuvem, com suas características, desafios, modelos, formas de entrega e de implantação.

O Capítulo 3, Arquiteturas para Sistemas de *Big Data*, mostra os trabalhos relacionados com arquiteturas para *big data* em nuvem pública ou privada, explicando as características, a motivação e os desafios enfrentados para implantantação das soluções

no modelo de IaaS. Assim sendo, são analisados ainda neste capítulo, os modelos, as técnicas e as tecnologias mais recentes.

O Capítulo 4, Arquitetura para BDaaS em Nuvem Privada, apresenta a proposta de arquitetura de BDaaS, com os seus componentes, as conexões entre esses componentes e os requisitos para atender a demanda para o processamento de dados em larga escala e as repectivas técnicas de construção. Assim, neste capítulo serão enfatizadas as tecnologias de implementação mais importantes, que são o Apache Hadoop [6] para o *big data*, e o OpenStack [104] para a nuvem privada.

O Capítulo 5, Conclusão, apresenta as ponderações sobre a proposta defendida neste trabalho, e as possíveis expansões da infraestrutura para tornar possível a inclusão de novos serviços.

Capítulo 2

Referencial Teórico

Neste capítulo são apresentados os conceitos, as características, os benefícios e os desafios da área da Computação em Nuvem e do *Big Data*. Ao final, são listadas as implementações e os *frameworks* que fazem parte do estado da arte destes campos de pesquisa.

2.1 Computação em Nuvem

A computação em nuvem, de acordo com Foster *et al.* [53], é um paradigma de computação distribuída em larga escala, conduzida por economias de escala, na qual um conjunto de recursos é entregue por demanda a usuários externos através da Internet. Este conjunto de recursos é formado por poder computacional, armazenamento, plataformas e serviços. A computação em nuvem difere de outros modelos por ser massivamente escalável, virtualizada, encapsulada em diferentes níveis de serviço para o cliente externo e por seus serviços serem configurados dinamicamente.

O estudo de Sotomayor *et al.* [117] tem a visão de que a computação em nuvem está ligada a questões de infraestrutura de recursos computacionais, tendo sua popularização a partir de 2006 com o serviço *Elastic Cloud Computing* (EC2) [3] da Amazon. Mesmo não sendo a primeira a propor o modelo, a empresa contribuiu para o modelo de computação utilitária, no qual o usuário pode provisionar máquinas virtuais no *datacenter* do provedor do serviço de nuvem, adicionando ou removendo capacidade da infraestrutura para atender à flutuação de sua demanda.

Outra definição para computação em nuvem é dada por Buyya *et al.* [24] e diz que a nuvem é um tipo de sistema paralelo e distribuído, consistindo de uma coleção de computadores interconectados e virtualizados que são tipicamente provisionados e apresentados como um ou mais recursos computacionais unificados, baseados em um SLA (*Service-Level Agreement*) estabelecido entre o provedor do serviço e os consumidores.

Além disso, segundo Mell e Grance [90], computação em nuvem é um modelo de compartilhamento de recursos computacionais incluindo rede, servidores, armazenamento, aplicações e serviços que podem ser provisionados e lançados rapidamente com mínimo esforço ou interação. Ademais, as características essenciais da computação em nuvem são [90]:

- Auto-serviço por demanda: um consumidor pode unilateralmente provisionar recursos computacionais, como tempo de servidor e armazenamento na rede sem necessidade de interação com o provedor do serviço;
- Amplo acesso à rede: as funcionalidades ficam disponíveis através da rede e podem ser acessadas pelas plataformas dos clientes;
- Agrupamento de recursos: os recursos computacionais são agrupados para servir a múltiplos clientes em um modelo *multi-tenant*. Os recursos são atribuídos dinamicamente de acordo com a demanda. O consumidor não tem controle ou conhecimento sobre a localização do recurso que está utilizando;
- Elasticidade rápida: a capacidade dos recursos computacionais pode ser rapidamente provisionada e liberada para escalar conforme a demanda;
- Serviço mensurável: o serviço de nuvem gerencia e otimiza os recursos, e tem a capacidade de monitoramento e controle de forma transparente para o provedor e para o consumidor do recurso.

Em Liu *et al.* [84] são definidos cinco atores principais para o modelo de computação em nuvem. Esses atores podem ser pessoas ou organizações que participam em transações, processos ou tarefas na computação em nuvem. Assim, os principais atores são:

- Consumidor da nuvem: uma pessoa ou organização que mantém um relacionamento de negócio e usa serviços do provedor de nuvem;
- Provedor de nuvem: uma pessoa, organização ou entidade responsável por tornar um serviço disponível para as partes interessadas;
- Auditor da nuvem: uma parte que pode conduzir avaliações independentes dos serviços da nuvem, operações nos sistemas de informação, performance e segurança da implementação de nuvem;
- Agente da nuvem (*Cloud Broker*): uma entidade que gerencia o uso, a performance e a entrega dos serviços da nuvem, e negocia o relacionamento entre os provedores e os consumidores da nuvem;

- Transportador da nuvem (*Cloud Carrier*): um intermediário que fornece conectividade e transporte dos serviços da nuvem entre provedores e consumidores.

2.1.1 Modelos de Serviços e Tipos de Nuvem

A respeito da forma como os serviços são entregues aos consumidores, chamados de modelos de serviços, existem as opções [90]:

- *Software as a Service* (SaaS): as aplicações são disponibilizadas para os clientes por meio do provedor de nuvem e ficam disponíveis via navegador ou outro tipo de interface, como um consumidor de *web services*;
- *Platform as a Service* (PaaS): as aplicações podem ser implantadas diretamente na infraestrutura da nuvem. Neste modelo, o cliente não tem controle sobre a infraestrutura de rede, de servidores ou de armazenamento, entretanto, ele tem controle sobre a configuração da plataforma na qual as aplicações são executadas;
- *Infrastructure as a Service* (IaaS): o cliente tem acesso direto à infraestrutura, que pode fazer o provisionamento de recursos fundamentais como processamento, armazenamento e rede. Neste modelo há também a liberdade para implantar o sistema operacional e as aplicações.

Em relação à forma de implantação, os tipos disponíveis, ainda segundo Mell e Grance [90] são:

- Nuvem Privada: a infraestrutura é provisionada exclusivamente para uma organização e seus consumidores;
- Nuvem Comunitária: a infraestrutura da nuvem é profisionada para uso exclusivo de uma comunidade de consumidores de organizações afins;
- Nuvem Pública: a infraestrutura é provisionada para uso aberto para o público em geral;
- Nuvem Híbrida: é a composição de dois ou mais tipos de infraestruturas distintas, a partir dos tipos apresentados anteriormente (privada, comunitária e pública).

2.1.2 Benefícios e Desafios

Conforme Mell e Grance [90], a computação em nuvem trouxe a facilidade do auto-serviço, amplo acesso à rede, agrupamento de recursos, elasticidade rápida e serviços mensurável. Dessa forma, novas ideias e produtos podem ser desenvolvidos sem o alto custo de equipamentos e mão de obra especializada, pagando apenas pelos recursos utilizados. Assim,

uma empresa pode começar a oferta do produto em pequena escala e, com o aumento da demanda, aumentar os recursos de hardware e de software de acordo com a sua necessidade.

Por outro lado, empresas com grandes sistemas podem utilizar a elasticidade da nuvem para escalar o processamento dos dados, o que pode ser feito com a utilização de uma nuvem pública, com a criação de uma nuvem privada ou a combinação destas duas (a nuvem híbrida).

Como visto em Armbrust *et al.* [20], a computação em nuvem tem relação com as aplicações (SaaS), a plataforma (PaaS) e a infraestrutura (IaaS) entregues como serviços para o usuário final. Dessa forma, os autores indicam que a computação em nuvem reduz os custos com eletricidade, rede, operações, software e hardware.

Deste modo, a computação em nuvem é adequada para situações nas quais a demanda por um serviço varia conforme o tempo, podendo haver picos em dias específicos do mês e ociosidade nos demais, assim, os recursos são redimensionados ou redirecionados automaticamente de acordo com a conveniência. A elasticidade é uma forma de diminuir o risco do superdimensionamento. É possível adicionar ou remover recursos em poucos minutos, adequando o processamento à carga de trabalho rapidamente. Todavia, há desafios a serem enfrentados, sendo que os principais [20]:

- Disponibilidade do serviço: espera-se que os serviços estejam sempre disponíveis, minimizandos o número de falhas para evitar prejuízos;
- Dados bloqueados: a interoperabilidade entre plataformas ainda está em estágio inicial e os usuários não podem facilmente transmitir dados entre elas. Uma solução seria a padronização das *Application Programming Interfaces* (APIs), de forma que os fornecedores de SaaS possam instalar seus produtos em diferentes plataformas de nuvem;
- Confidencialidade/auditabilidade dos dados: segurança é uma questão a ser levantada durante a adoção do modelo de nuvem pública, na qual a responsabilidade é dividida entre o usuário, o fornecedor do serviço de nuvem e outros fornecedores envolvidos no processo;
- Gargalos na transferência de dados: o custo para transferência de grandes volumes de dados na Internet é alto, fato que deve ser considerado porque as aplicações podem usar mais de um provedor de nuvem para garantir redundância;
- Performance imprevisível: a CPU e a memória de máquinas virtuais (VMs) são compartilhadas com eficiência na nuvem. Contudo, a performance de I/O de disco

e rede virtualizados é problemática. Uma possível solução seria a utilização de *Solid-state drive* (SSD) [20];

- Elasticidade rápida: a nuvem deve aumentar e diminuir rapidamente a quantidade dos recursos computacionais. Uma opção, segundo [20], é usar *machine learning* como ferramenta de predição;
- Reputação: a reputação do provedor de nuvem pode ser prejudicada se um dos clientes tiver mal comportamento;
- Licenciamento de software: o modelo de licenciamento de software comercial não é o mais indicado para os provedores de nuvem, o que representa uma oportunidade para o software grátis e livre.

De acordo com Kambatla *et al.* [76], os *datacenters* são baseados em computação e armazenamento virtualizado e, tipicamente, escalam para milhares de VMs, podendo se espalhar para outros *datacenters*. Este ambiente enfrenta desafios, como a análise de *big data*, a diversidade e a escala das aplicações. Assim, verifica-se que há uma associação entre o crescimento do volume de dados, a complexidade das análises e o aumento do número de sistemas criados para atender às demandas.

Ainda em [76], e considerando uma carga de trabalho intensa, foram definidas dimensões que incluem o tempo de resposta, o padrão de acesso, o tipo de dados, a velocidade de I/O e a complexidade do trabalho. A partir destas dimensões foram definidas as situações atuais e futuras nas quais a análise de *big data* pode ser utilizada, que são: (i) ordenação distribuída na escala de *petabytes*, (ii) busca indexada em memória, (iii) sistemas de recomendação, (iv) deduplicação de dados baseados em acesso não sequencial, e (v) processamento de vídeo em tempo real de forma interativa com o usuário.

O artigo de Zhang *et al.* [130] lista outros desafios da computação em nuvem. Segundo os autores, apesar da ampla adoção da computação em nuvem pela indústria, restam ainda questões que não estão totalmente resolvidas. Os desafios citados por Zhang *et al.* [130] que se relacionam com esta dissertação são:

- Provisionamento automatizado de serviços: a alocação e a liberação de recursos é o ponto chave da computação em nuvem, mas mapear requisitos de serviço (por exemplo QoS) em requisitos de baixo nível (por exemplo CPU) não é trivial;
- Consolidação de servidores: técnica na qual as VMs são agrupadas para maximizar a utilização do servidor, enquanto diminui o consumo de energia;
- Segurança dos dados: as políticas de segurança são de responsabilidade do provedor de nuvem, e precisam garantir confidencialidade e auditabilidade. Por isso, é necessária a criação de mecanismos de confiança na infraestrutura da nuvem;

- *Frameworks* de software: a performance de aplicações que usam *frameworks* como o MapReduce é fortemente dependente do tipo de operação, por exemplo, ordenação demanda o sistema de I/O, enquanto que análise de texto usa primariamente CPU. Assim, verifica-se que os *frameworks* apresentam gargalos de performance que podem ser otimizados com configurações dinâmicas;
- Tecnologias de armazenamento e gerenciamento de dados: os *frameworks*, incluindo Hadoop, são projetados para operar com diversos sistemas de arquivos e fontes de dados, entretanto, ainda há problemas na interoperabilidade com os sistemas de arquivos legados.

Assim sendo, a Seção 2.1.3 detalhará o modelo de *Infrastructure as a Service*, que será usado como base em diversos pontos desta dissertação.

2.1.3 Infraestrutura como Serviço (IaaS)

O estudo de Dukaric e Juric [44] considera que o IaaS é uma das camadas mais importantes da computação em nuvem, e que não há mecanismos eficientes para analisar, comparar e avaliar suas implementações. Assim, os autores propõem um *framework* para arquitetura de IaaS, dividido em sete camadas:

- Abstração de recursos: tem papel crucial na infraestrutura de nuvem, e compreende os componentes de computação, armazenamento, volume e rede;
- Núcleo de serviços: principal parte da infraestrutura proposta, inclui os serviços de identificação, agendamento, repositório de imagens, cobrança, faturamento e *logging*;
- Suporte: camada de *middleware* que fornece os meios para que as outras camadas se comuniquem. Dispõe de um barramento de mensagens, banco de dados para armazenamento do estado da infraestrutura e serviço de transferência;
- Segurança: fornece serviços de autenticação, autorização e monitoramento;
- Gerenciamento: compreende as ferramentas de gerenciamento, federação, elasticidade, energia etc. Além disso, especifica APIs, ferramentas, interfaces gráficas e orquestração para interação entre serviços e recursos;
- Controle: inclui os serviços de SLA, medição, sistema de políticas, notificação e orquestração;
- Serviços de valor agregado: gerencia os componentes complementares, como gerenciamento de disponibilidade de zonas, alta disponibilidade, suporte a nuvem híbrida, migração ao vivo de VMs (*live migration*) etc.

O trabalho de Sefraoui *et al.* [115] mostra um estudo comparativo de ferramentas de IaaS, com uma arquitetura de hospedagem em nuvem escalável e *open source*. A solução apresenta o OpenStack [104], detalhando sua arquitetura e funcionalidades, mas também faz um comparativo com outros dois softwares para implementação de nuvens: Eucalyptus [48] e OpenNebula [100]. Neste estudo, os autores concluem que o OpenStack é a melhor opção para soluções baseadas em *open source* e com suporte para novos padrões.

2.2 Nuvem Privada

De acordo com Mell e Grance [90], a nuvem privada é um das formas de implantação da computação em nuvem, na qual os recursos computacionais estão disponíveis apenas para uma organização e seus consumidores. Ela pode pertencer, ser gerenciada, localizada e operada pela própria organização, por uma empresa terceirizada ou ainda algum tipo de combinação entre elas.

Segundo Assunção *et al.* [21], uma nuvem privada é aquela implantada em uma rede privada, gerenciada pela organização ou por uma empresa terceirizada. Ela é adequada para negócios que precisam de alto nível de controle de segurança e privacidade de dados. Considerando este contexto, a nuvem privada pode ser usada para oferecer serviços e dados eficientemente entre diferentes departamentos de uma grande empresa.

Conforme Armbrust *et al.* [20], uma nuvem privada se refere a *datacenters* de empresas ou organizações que não estão disponíveis para o público em geral. Neste caso, a infraestrutura é suficiente para se beneficiar com as vantagens da computação em nuvem.

Como visto em Vogel *et al.* [123], do ponto de vista da infraestrutura, uma nuvem IaaS é dividida em quatro camadas, que foram usadas como base para comparar e analisar a flexibilidade e a resiliência do OpenStack [104], do CloudStack [33] e do OpenNebula [100]. As camadas são:

- Gerenciador de nuvem: nível mais alto da infraestrutura, controla os usuários, os grupos, as permissões, as quotas e a qualidade do serviço;
- Gerenciador de infraestrutura: responsável pelo agendamento de recursos, imagens, rede, volumes, *templates* e criação das VMs;
- Gerenciador de máquinas virtuais: fornece uma abstração dos recursos físicos. É implementado através de um *hypervisor* que é executado no sistema operacional ou diretamente no servidor (*bare metal*);
- Hardware: compreende o nível mais básico da infraestrutura, com os recursos computacionais físicos.

2.2.1 Características e Vantagens

Muitas características das nuvens privadas e públicas são similares [20] e incluem o auto-serviço por demanda, o amplo acesso à rede, o agrupamento de recursos, a elasticidade rápida e o serviço mensurável. No entanto, há diferenças no nível de segurança, privacidade e gerenciamento dos recursos [67].

Jadeja e Modi [67] dizem que a maior vantagem da nuvem privada é a sua facilidade para o gerenciamento da segurança, manutenção, atualização, além de permitir maior controle sobre sua implantação e uso. Uma vez que a nuvem privada opera dentro do *datacenter* da própria organização, ela pode ser comparada à intranet. Diferentemente da nuvem pública, na qual os serviços são gerenciados pelo provedor do serviço, ou seja, na nuvem privada os recursos são reunidos e gerenciados pela própria organização. Dessa forma, os recursos e as aplicações são disponibilizados apenas para os usuários da organização, reforçando a segurança.

Em Hashem *et al.* [61], tem-se que a nuvem privada oferece melhor nível de privacidade para os dados da organização. A privacidade continua sendo uma preocupação, dificultando a utilização da nuvem pública. Além disso, é importante considerar também o cenário no qual o uso de ferramentas de *big data* para mineração e análise de dados revela informações pessoais, produzindo resultados significativos como serviços personalizados e baseados em localização. Neste contexto, a exposição desses dados sensíveis levanta questões sobre roubo e perda de dados.

2.2.2 Desafios

Os desafios enfrentados na área de nuvem privada são referentes à segurança e implantação da infraestrutura. Como visto em Horey *et al.* [63], os autores reportam que a implantação da infraestrutura de nuvem privada é uma preocupação, principalmente, para serviços distribuídos complexos.

Assim, considerando apenas a questão da segurança, a adoção de nuvem privada demanda análise cuidadosa da infraestrutura atual e suas necessidades. Neste caso, o cenário mais indicado para a escolha de uma nuvem privada é aquele no qual a empresa tem dados sensíveis.

A dificuldade de implantação é compartilhada com a área de *big data*, como reportado por Kart *et al.* [77] em uma pesquisa com 720 executivos de organizações de TI, na qual 29% das empresas consideram que a configuração e a manutenção da infraestrutura para *big data* é uma atividade complicada, dificultando a entrega de soluções devido à falta de mão de obra especializada.

2.2.3 Ferramentas de Nuvem Privada

Várias pesquisas têm sido feitas sobre a avaliação de técnicas e de ferramentas para a implantação de nuvem privada, mais notadamente utilizando o Apache CloudStack [33], o OpenNebula [100] e o OpenStack [104].

As funcionalidades e a performance dessas ferramentas são semelhantes [123], contudo, o OpenStack apresenta módulos especializados em Hadoop, o que tornam esta plataforma mais adequada ao BDaaS. Assim, as próximas seções descrevem detalhadamente essas ferramentas.

CloudStack

O Apache CloudStack [33] é um software para implantação de plataforma de nuvem IaaS. Ele é usado para oferecer serviços de nuvem pública, privada ou híbrida, haja vista que sua API é compatível com o Amazon EC2 [3] e o Amazon S3 [4]. Atualmente, ele suporta os *hypervisors* mais usados na indústria, incluindo o VMware [122], KVM [80], XenServer [127], Xen Cloud Platform [126], Oracle VM [109] e Hyper-V [93].

A ferramenta oferece uma solução completa com serviços de orquestração de computação, gerenciamento de rede, armazenamento, *multi-tenancy* e gerenciamento de usuários, fornecendo uma plataforma confiável e escalável de nuvem privada ou pública. O acesso aos serviços administrativos pode ser feito por meio de interface gráfica, APIs nativas e interface de linha de comando [33]. Conceitualmente, o CloudStack é construído sobre pilhas de serviços e diversos agentes para controlar a nuvem, o armazenamento distribuído, as imagens e a rede.

OpenNebula

O OpenNebula [100] é uma solução para gerenciamento de *datacenters* virtualizados para fornecimento de nuvens IaaS privadas, públicas ou híbridas, disponibilizando funcionalidades que atendem tanto a virtualização quanto a infraestrutura da nuvem. Ele permite a consolidação de servidores, *multi-tenancy*, integração dos recursos computacionais para computação, armazenamento e conectividade. A ferramenta suporta os principais *hypervisors*, incluindo KVM [80], Xen[126] e VMware ESX [122].

O software usa conceitos e nomenclatura similares aos sistemas Linux, integrando serviços baseados neste sistema operacional, como *scripts* e componentes. Outro conceito importante presente no OpenNebula é que sua infraestrutura é formada por um servidor *master* e vários servidores *slave*. Dessa forma, o *master* funciona como *front-end* da nuvem, no qual são configurados os serviços de máquinas virtuais, transferência de dados, gerenciamento de rede etc.

OpenStack

O OpenStack é uma ferramenta de IaaS escalável, amplamente usada e testada com empresas nas quais os dados alcançam *petabytes*, gerenciando milhões de máquinas virtuais [115]. O código do OpenStack é aberto e suporta as mais importantes soluções de virtualização como KVM [80], Hyper-V [93], LXC [83], QEMU [111], Xen [126] e XenServer [127]. A estrutura do OpenStack é modular e compreende, essencialmente, as seguintes funcionalidades:

- Servidor de API: interface de administração que controla o *hypervisor*;
- *Message queue*: envia as instruções e facilita a comunicação entre os componentes;
- Controlador de processamento: gerencia o ciclo de vida dos servidores virtuais;
- *Object store*: responsável pelo serviço de armazenamento;
- Controlador de volumes: gerencia a manipulação dos volumes;
- Controlador de rede: gerencia *bridges*, VLAN, DHCP, DNS e regras do *firewall*;
- Orquestrador: distribui as tarefas e determina onde elas devem ser executadas.

O software foi construído sobre uma pilha de APIs que são usadas para comunicação entre os diversos módulos e serviços. Esta divisão em módulos permite a escalabilidade da ferramenta. Um desses módulos é o projeto Sahara [101], que é uma ferramenta para provisionar *clusters* Hadoop na forma de PaaS no OpenStack. Ele suporta a execução de programas em *clusters* existentes ou através de *clusters* criados dinamicamente. É possível especificar parâmetros de configuração como a versão do Hadoop, a topologia do *cluster*, o hardware dos nós etc. Assim, a implantação de *clusters* Hadoop é simplificada pela utilização de *templates* pré-configurados [95].

Apesar de todas as evoluções tecnológicas, a complexidade da configuração e da manutenção dos ambientes de *big data*, e de computação em nuvem continua sendo um desafio para as empresas. Dessa maneira, esta realidade motivou a criação de uma nova forma de entregar serviços na nuvem, chamada de *Big Data as a Service*, ou BDaaS.

Com o BDaaS, as empresas podem se mover rapidamente para a fase de implantação de soluções de *big data* sem passar pelas fases iniciais de pesquisa e de avaliação de ferramentas. Contudo, antes da implantação, a arquitetura deve estar documentada para permitir o reuso de seus componentes. A formalização da proposta de arquitetura do BDaaS será feita por meio de *frameworks* e padrões internacionais, detalhados no próximo capítulo.

2.3 Big Data

O termo *big data* tem várias definições, de acordo com o contexto a que se aplica. Uma das definições apresentadas é de Chang [27] e diz que *big data* consiste de *datasets* com as características de volume, variedade, velocidade e/ou variabilidade que requerem um arquitetura escalável para armazenamento, manipulação e análise eficiente. Ainda em Chang [27], *big data* se refere à inabilidade das arquiteturas de dados tradicionais de manipularem eficientemente os novos *datasets*, forçando a criação de novas arquiteturas, que consistem de sistemas de dados distribuídos em recursos computacionais independentes, acoplados horizontalmente para alcançar a escalabilidade, utilizando processamento massivamente paralelo.

Historicamente, as análises eram feitas em dados estruturados com o modelo de dados relacional. Atualmente, o *big data* trouxe os dados não estruturados e os semi-estruturados, na forma de texto, micro-texto, páginas web, dados com relacionamento, e *streaming* de imagens e vídeos. Este é o principal valor gerado pela análise de *big data*, ou seja, a capacidade de processar grandes volumes de dados, vários tipos de informação e em uma alta velocidade. Assim, além do volume massivo de dados, a análise de *big data* traz técnicas eficientes de processamento [27].

Com isso, Chang [27] considera que o conceito de *big data* não é trivial e deve ser claramente definido. Assim, uma solução *big data* deve considerar as características dos conjuntos de dados (*datasets*), o tipo de análises que serão feitas, a performance do sistema e as considerações de custo/benefício do negócio. Neste sentido, são características do *big data* [27]:

- Volume: o tamanho do *dataset* é grande, da ordem de milhões, bilhões ou trilhões de registros;
- Variedade: os dados provém de múltiplas fontes, domínios e tipos;
- Velocidade: ritmo rápido de geração dos dados;
- Variabilidade: mudanças constantes nas características dos dados.

Esses aspectos são conhecidos como os quatro 'Vs' do *big data*. Mesmo que outros 'Vs' tenham sido indicados com o passar do tempo, esses quatro itens são responsáveis por direcionar o desenvolvimento das novas arquiteturas de processamento, análise e armazenamento de dados [27].

Em Uddin *et al.* [121] é mostrada outra caracterização que considera sete 'Vs' para mostrar o potencial de solução de problemas reais proporcionado pelo *big data*. Assim, segundo o artigo [121], as propriedades do *big data* são:

1. Volume: refere-se ao tamanho dos dados sendo gerados pelas diversas fontes;
2. Velocidade: refere-se à alta velocidade com que os dados são gerados e movidos para o dispositivo de armazenamento para posterior análise;
3. Variedade: trata dos vários formatos de dados, incluindo audio, texto, imagens etc;
4. Veracidade: refere-se à certeza de que este dado está correto e pode contribuir para um determinado cenário;
5. Validade: refere-se à correção e precisão dos dados em relação ao uso pretendido. Um dado pode ser útil para um contexto e ao mesmo tempo pode ser inútil em uma aplicação diferente;
6. Volatilidade: refere-se à política de retenção dos dados ao longo do tempo;
7. Valor: é o resultado esperado para uma aplicação de *big data*. Este valor deve superar o custo necessário para seu processamento e gerenciamento.

Em Apache Software Foundation [6], *big data* é definido como *datasets* que não poderiam ser capturados, gerenciados e processados em computadores comuns dentro de um escopo aceitável. A partir dessa caracterização, Manyika *et al.* [87] aponta que *big data* é a próxima fronteira para inovação, competitividade e produtividade. Assim, *big data* é definido pelos autores como sendo *datasets* que crescem em escala ao longo do tempo e que não podem ser adquiridos, armazenados e gerenciados pelos bancos de dados tradicionais. Dessa forma, o volume dos *datasets* não é o único critério para definir uma solução *big data*.

Como visto em Gantz e Reinsel [54], as pesquisas sobre *big data* descrevem uma nova geração de tecnologias e arquiteturas, projetadas para extrair economicamente valor de grandes volumes de uma ampla varidade de dados, habilitando a captura em alta velocidade, descoberta de conhecimento e análises. Com isso, foi criada uma classificação, cujo objetivo é explicar as propriedades do *big data*. Essa classificação é mostrada na Figura 2.1.

É possível notar na Figura 2.1 que cada categoria tem especificações diferentes usadas para selecionar as tecnologias mais adequadas para a construção da solução de *big data*. Isto posto, há cinco subgrupos associados aos dados, os quais são [61]:

- Fontes: web, rede social, máquinas, sensores, transações e Internet *of Things* (IoT);
- Formatos de conteúdo: estruturado, não estruturado e semi-estruturado;
- Armazenamento de dados (*data stores*): os bancos NoSQL são classificados em orientados a documento, orientados a coluna, baseado em grafo e chave-valor;

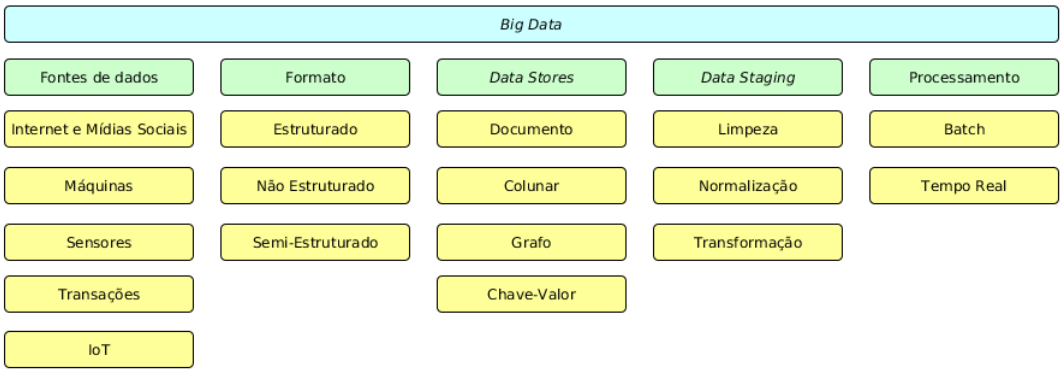

Figura 2.1: Classificação para *big data*. Adaptado de [61].

- *Data staging*: estágio intermediário de armazenamento para realização de tarefas que incluem a limpeza, a normalização e a transformação dos dados;
- Processamento: pode ser realizado em lote (*batch*) e em tempo real (*real time*).

Segundo Chang [27], essa explosão de dados gera novas oportunidades para combinar fontes de informação com o objetivo de gerar valor. Em contraste com os sistemas de dados tradicionais, a análise de *big data* trouxe uma mudança significativa em termos de arquitetura para suportar o volume de processamento demandado, bem como em relação ao tipo de informação.

De acordo com Hopkins *et al.* [62], o uso de *Business Intelligence* (BI) é adequado até o ponto no qual o volume ou a variedade dos dados começa a crescer. A partir de então, uma solução *big data* tende a ser mais adequada. Ainda em Hopkins *et al.* [62], pode-se verificar que o uso de soluções de *big data* pode entregar valor com menor custo e em menor espaço de tempo, como visto na Figura 2.2.

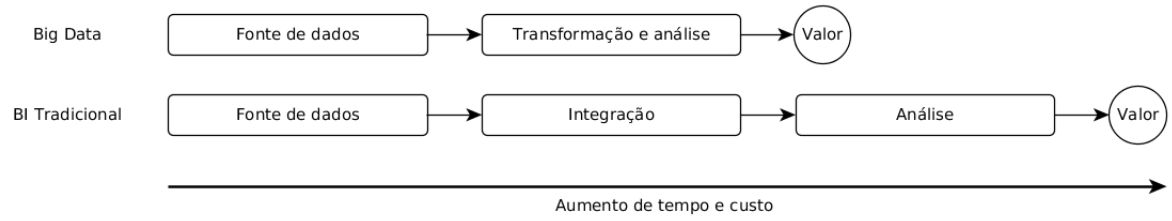

Figura 2.2: Relação entre *big data* e BI considerando tempo e custo. Adaptado de [62].

2.3.1 Fases dos Sistemas de *Big Data*

Um sistema de *big data* é composto de funcionalidades para tratar as diferentes fases do ciclo de vida dos dados, desde o nascimento até seu descarte. Do ponto de vista da engenharia de sistemas, pode-se decompor um sistema de *big data* em quatro fases

consecutivas, sendo elas a geração, a aquisição, o armazenamento e a análise dos dados, como listado em Hu *et al.* [65]. Assim, essas fases são:

- **Geração de dados**: diz respeito à forma como os dados são gerados, considerando que existem fontes distribuídas e complexas, como sensores e vídeos. Nesta dissertação o caso de uso de validação usa os dados gerados por sistemas corporativos do Governo Federal, especificamente, os dados de processos judiciais. Entretanto, há desafios técnicos para coletar este tipo de informação, seja por sua complexidade, seja por sua quantidade, o que levanta a necessidade do uso de novas tecnologias para atender à demanda;
- **Aquisição de dados**: é o processo de obtenção dos dados. Este processo inclui o particionamento em coleções específicas, a transmissão e o pré-processamento dos dados. Considerando as origens e os formatos dos dados, o passo inicial é a coleta e o particionamento dos dados em coleções específicas. Em seguida, os dados devem ser transferidos em alta velocidade para o dispositivo final de armazenamento, que será fonte para as diversas aplicações clientes. Para finalizar, são realizadas operações de *data cleansing* [65], uma técnica para verificar se há erros nos dados de entrada e, então, fazer a correção ou o descarte dos registros;
- **Armazenamento de dados** se refere às capacidades de retenção e gerenciamento de *datasets*. O sistema de armazenamento é formado pela infraestrutura de hardware e de software para gerenciamento. Considerando o objetivo desta dissertação, nesta fase serão utilizadas tecnologias de computação em nuvem para permitir a escalabilidade, a configuração automática e as interfaces de acesso, características necessárias para atender a diferentes tipos de aplicações.
- **Análise de dados**: apresenta novos métodos e ferramentas para consultar e extrair informações dos *datasets*. Essas ferramentas são usadas para analisar dados estruturados, dados em formato textual, dados de multimídia, dados de redes sociais e de dispositivos móveis. O caso de uso a ser apresentado neste trabalho contém primariamente dados estruturados, com uma parcela menor de dados semi-estruturados.

Neste trabalho é acrescentada a fase de **integração de dados**, para que os sistemas corporativos possam consumir os dados por meio de *web services*, uma abordagem que garante baixo acoplamento entre o sistema de *big data* e os consumidores de dados.

2.3.2 Big Data as a Service (BDaaS)

O *Big Data as a Service* (BDaaS) é um modelo que combina as facilidades da computação em nuvem com o poder de processamento dos sistemas de *big data* para entregar serviços

de dados, de banco de dados, de análise de dados e de plataforma de processamento. O BDaaS é um modelo relativamente novo de serviço em nuvem para provisionamento de recursos computacionais relacionados com sistemas de *big data*, além de usar os modelos de serviço tradicionais da computação em nuvem (Paas, SaaS e IaaS) [128].

O BDaaS funciona como uma camada de abstração acima dos serviços de dados, de forma que o usuário seleciona a funcionalidade, e a infraestrutura subjacente se encarrega de provisionar, instalar e configurar os serviços, que são tarefas complexas que exigem mão de obra especializada. Assim, neste modelo é possível implantar rapidamente sistemas de *big data*, reduzindo o tempo e o custo do desenvolvimento nas etapas iniciais do ciclo de vida do projeto [63, 131].

A implementação de sistemas *big data* envolve altos custos para configuração da infraestrutura e obtenção de mão de obra especializada. Assim, uma arquitetura de BDaaS pode ajudar as organizações a se moverem rapidamente do ponto inicial, que é a pesquisa de tecnologias de *big data*, para a fase final de implantação da solução. Desta forma, mesmo as fases que envolvem projetos-piloto seriam agilizadas com o uso de tecnologias baseadas em nuvem [23].

O BDaaS compreende outros modelos de serviço específicos para endereçar as demandas dos sistemas de *big data*, ou seja, o BDaaS é o resultado da combinação dos modelos de serviço tradicionais (PaaS, SaaS e IaaS) com as funcionalidades próprias dos sistemas de *big data*, gerando novos modelos de serviço. Assim, para caracterizar o BDaaS, a infraestrutura de nuvem deve oferecer ao menos um dos seguintes modelos de serviço:

- **Data as a Service** (DaaS): refere-se à disponibilização de conjuntos de dados por meio de *web services*. Esta abordagem é agnóstica em relação à linguagem de programação ou tecnologias adotadas pelos sistemas externos. É implementado por meio de *Application Programming Interface* (API) que permite as operações de inclusão, alteração, exclusão e consultas básicas pré-definidas, entretanto, não suporta consultas SQL. Os serviços de dados são independentes entre si e reusáveis. Os registros costumam estar armazenados em NoSQL, mas pode acessar também o sistema de arquivos e executar *jobs* previamente configurados;
- **Database as a Service** (DBaaS): refere-se ao provisionamento de bancos de dados NoSQL. Apesar de ser tecnicamente possível, a arquitetura proposta neste trabalho não provisiona bancos de dados relacionais, limitando-se a sistemas de *big data*;
- **Big Data Platform as a Service** (BDPaaS): refere-se ao provisionamento de *clusters* de *big data* para executar programas Hadoop ou Spark[1]. A plataforma de nuvem privada é responsável pela instalação e pela configuração do software,

[1]https://spark.apache.org/

diminuindo o problema da complexidade do gerenciamento dos ambientes de *big data*;

- **Analytics as a Service** (AaaS): refere-se ao provisionamento de ferramentas de análise de dados a partir de *clusters* de *big data*;
- **Storage as a Service** (StaaS): refere-se ao provisionamento de armazenamento por meio de sistemas de arquivos distribuídos, do inglês *distributed file systems* (DFS).

Grandes provedores de nuvem como Amazon, Microsoft e Oracle já oferecem o modelo de BDaaS em nuvem. Nestes casos, o provedor é responsável pelos equipamentos, pela operação do *datacenter* e pelos serviços de *big data*.

2.3.3 Tipos de Processamento de Dados

Segundo Chang [26], os *frameworks* de processamento para *big data* fornecem a infraestrutura de software necessária para suportar o volume, a velocidade, a variedade e a variabilidade das aplicações. Com isso, são necessárias várias plataformas e tecnologias para atender aos diversos desafios da análise de dados.

Na visão de Hashem *et al.* [61] existem dois tipos de processamento de dados: *batch* e tempo real (*real-time*). Essas categorias são avaliadas de acordo com a velocidade com que o usuário recebe a resposta. No caso do processamento *batch*, existe alta latência e o usuário precisa esperar minutos ou mesmo horas pelo resultado. Este tipo de processamento é usado em cenários nos quais são feitas análises de séries históricas de dados, usando todo ou grande parte de um *dataset*.

O processamento em *real time* tem baixa latência e o usuário recebe o resultado em questão de segundos. Esta situação é também chamada de *Complex Event Processing* (CEP), processamento em tempo quase real (*near real-time*) ou mesmo processamento de *streaming* e se refere ao processamento de fluxos constantes de dados, que são inseridos no sistema pelos usuários externos das aplicações. Para seu uso efetivo, o processamento em *real time* tem duas importantes considerações [23]:

- Armazenamento: o sistema de armazenamento deve ser capaz de persistir o fluxo de dados, além de permitir sua análise. O Hadoop Distributed File System (HDFS) [9], por exemplo, permite o armazenamento distribuído e tem altas taxas de transferência de dados;
- Resposta: em determinadas situações, a resposta às requisições do usuário deve ser imediata. Para armazenar, analisar e processar dados em *real time* são utilizadas ferramentas como HBase [8] e Cassandra [5].

Neste contexto, um dos diversos *frameworks* para processamento de *big data* é o Apache Hadoop [6], que se tornou, de fato, um dos mais amplamente usados. Inicialmente projetado para processamento *batch*, o Hadoop evoluiu para um completo ecossistema de projetos com diferentes funcionalidades e tecnologias, os quais serão vistos em detalhe na Seção 2.4.

2.4 Apache Hadoop

O Apache Hadoop [6] é um *framework open source* para processamento distribuído de grandes volumes de dados por meio de *clusters* de computadores de baixo custo. É uma solução bastante usada na computação em nuvem e aplicações *big data* [23, 61].

Inicialmente, o Hadoop era baseado apenas no HDFS e no MapReduce [11]. Entretanto, devido a sua flexibilidade de uso, o *framework* evoluiu para além do processamento *batch*, de forma que o termo Hadoop tem sido usado como sinônimo de um ecossistema de ferramentas de infraestrutura para computação distribuída, e processamento de dados em larga escala [125].

Os casos de uso do ecossistema Hadoop incluem situações diversas, como mecanismos distribuídos para consultas SQL interativas em grandes bases de dados, processamento iterativo em algoritmos de *machine learning*, processamento de dados em fluxo constante (*streaming*) e sistemas de busca textual [125]. Internamente, o Hadoop é composto por quatro módulos [6], os quais são:

- *Hadoop MapReduce*: um sistema para processamento paralelo de *big data*;
- *Hadoop Distributed File System* (HDFS): sistema de arquivos distribuído de alta performance;
- *Hadoop Common*: programas utilitários e bibliotecas de suporte do projeto;
- Hadoop YARN (*Yet Another Resource Negotiator*): *framework* para gerenciamento de recursos do *cluster*.

Cada um desses módulos será descrito em detalhes nas próximas seções.

2.4.1 Hadoop MapReduce

O Hadoop MapReduce é um sistema de processamento em *batch* e um modelo de programação para escrever aplicações distribuídas que acessam grande volumes de dados em paralelo [11, 125]. O MapReduce suporta *clusters* com tolerância a falhas rodando em milhares de nós com hardware de baixo custo. Além disso, o MapReduce fornece um

modelo de programação simplificado que abstrai a complexidade de leitura e de escrita (I/O) de dados em ambientes distribuídos [125].

De acordo com Dean e Ghemawat [42], o MapReduce é adequado para processamento e geração de grandes *datasets*. Programas MapReduce são automaticamente paralelizáveis e executados em grandes *clusters* de máquinas de baixo custo. O MapReduce é o responsável pelo sistema de processamento paralelo e distribuído, e implementa funcionalidades como particionamento dos dados, agendamento das execuções através do conjunto de máquinas, manipulação das falhas e comunicação entre nós do *cluster*. Assim, o *framework* é adequado para diversos cenários, os principais são [42]:

- *Grep* distribuído: emite uma linha quando encontra determinado padrão de texto;
- Contagem da frequência de acesso a URL: análise do *log* das requisições de acesso a páginas web, e retorna o número de vezes que a URL foi acessada;
- Grafo de *link* reverso (*Reverse Web-Link Graph*): recupera a lista de URLs de origem associadas a uma URL de destino;
- Vetor de termos por *host* (*Term-Vector per Host*): retorna um vetor de termos com a lista de palavras mais frequentes em uma lista de documentos;
- Índice invertido: principal funcionalidade de um mecanismo de busca, o índice invertido retorna um conjunto de pares chave-valor no formato *palavra, lista(identificador do documento)*, no qual cada palavra está associada aos documentos em que ela aparece;
- Ordenação distribuída: o programa extrai a chave de cada registro e, com base nela, distribui o par chave-valor no formato *chave,registro* entre os nós do *cluster* para ordenação.

Conforme White [125], o MapReduce não é adequado para análise interativa. As aplicações MapReduce, geralmente, executam em minutos ou horas, tornando-o apropriado para análise *off-line*, aquela na qual o usuário não fica aguardando o resultado em frente ao computador.

Um programa MapReduce, como o nome sugere, envolve as tarefas (ou funções) *Map* e *Reduce*. Durante a fase *Map*, os dados são lidos de sua origem, geralmente o HDFS, de forma paralela. Neste mapeamento é feita a seleção dos dados que serão analisados na sequência. O *framework* ordena esse resultado intermediário que será usado como fonte de dados para a fase *Reduce*. Por fim, durante a fase *Reduce* é executada uma função de agregação (redução) [11].

Tanto o MapReduce quanto o SGBDR podem ser usados para análise de dados, contudo, de acordo com Lam [81], o MapReduce tem a vantagem de não se limitar a dados

estruturados, enquanto que os sistemas que suportam SQL foram projetado para dados tabulares. Isto posto, para justificar sua adoção, é comum a comparação entre Hadoop e SGBDR.

Um SGBDR é uma boa alternativa quando a maioria das operações envolve consultas aleatórias ou atualização, na qual a solução demanda baixa latência e o tamanho dos dados está na escala de *gigabytes*. Quando a análise de dados pode ser realizada em *batch* e precisa ler o *dataset* completo, então o MapReduce tende a ser mais performático. De forma geral, o MapReduce não é considerado um substituto para o SGBDR, mas sim um complemento [81, 125]. Há, inclusive, implementação de SQL em cima do Hadoop. A Tabela 2.1 mostra os pontos que podem ser comparados entre o MapReduce e o SGBDR [125].

Tabela 2.1: Comparativo entre SGBDR e MapReduce. Adaptado de [125].

Característica	**SGBDR**	**MapReduce**
Tamanho dos dados	*Gigabytes*	*Petabytes*
Acesso	Interativo e *batch*	*Batch*
Atualizações	Leitura e gravação repetidamente (*Read and write many times*)	Uma única gravação e várias leituras (*Write once, read many times*)
Transações	ACID	Não há
Estrutura	Definida antes da gravação (*Schema-on-write*)	Definida no momento da leitura (*Schema-on-read*)
Integridade	Alta	Baixa
Escalabilidade	Não-linear	Linear

O funcionamento do MapReduce é baseado na ideia de pares chave-valor, enquanto que no SGBDR os dados residem em tabelas estruturadas em esquemas. O MapReduce é adequado para as aplicações modernas, que tratam informações que não se encaixam bem nos bancos de dados relacionais, como documentos de texto, imagens, arquivos XML, JSON, CSV, TSV etc. Na verdade, o MapReduce pode trabalhar virtualmente com qualquer formato de dados. Entretanto, internamente, ele fará a transformação em pares chave-valor [42, 81].

Contudo, é importante ressaltar que, como reportado por Appuswamy *et al.* [19], o Hadoop foi projetado para escalar horizontalmente (*scale-out*), aproveitando os *clusters* de baixo custo. O entendimento comum tanto da indústria quanto da academia é que esta estratégia é melhor do que escalar verticalmente (*scale-up*), com a adição de mais recursos computacionais em um único servidor de alto custo. Entretanto, a maioria das análises com Hadoop processam menos de 100 GB de dados, e segundo o estudo de Appuswamy *et al.* [19], para este tipo de análise, o uso de *scale-up* (elasticidade vertical) pode ser igual

ou melhor que *scale-out* (elasticidade horizontal), ou seja, o uso de um único servidor de alto custo é semelhante a usar um *cluster* com várias máquinas.

Na mesma direção, Li *et al.* [82] dizem que máquinas em *scale-up* são mais indicadas para processamentos pequenos ou medianos, que envolvem a escala de *kilobytes* ou *megabytes*. Para volumes de dados que envolvem *gigabytes* ou *terabytes*, máquinas em *scale-out* são mais adequadas. Uma vez que as cargas de trabalho costumam ter diferentes tamanhos de dados, uma abordagem híbrida pode ser a melhor alternativa.

Em Hashem *et al.* [61], a computação em nuvem e o *big data* estão fortemente relacionados. Uma solução *big data* tem a capacidade de usar a computação distribuída para processar com velocidade, enquanto a computação em nuvem fornece a infraestrutura (IaaS) para rodar os programas Hadoop. Assim, as fontes de dados da nuvem podem ser armazenadas em bases de dados distribuídas e tolerantes a falha.

Ainda de acordo com Hashem *et al.* [61], a infraestrutura da nuvem fornece uma plataforma de armazenamento eficiente para análise de *big data*, com isso, a computação em nuvem funciona como um modelo de serviço através do processamento de *big data*.

2.4.2 Hadoop Common

O Hadoop Common é um conjunto de bibliotecas e utilitários que oferecem suporte aos demais módulos do Hadoop. Ele representa uma camada de abstração entre o Hadoop e o sistema operacional, disponibilizando serviços básicos de administração do *cluster* através de interface de linha de comando [6].

Anteriormente chamado de Hadoop Core, o Hadoop Common contém os arquivos e os *scripts* necessários para executar o *cluster* Hadoop, bem como o código-fonte e a documentação [7]. Ele é composto por serviços de autorização, autenticação, configuração do *cluster*, métricas etc.

2.4.3 Hadoop YARN

A partir da versão 2, o Hadoop conta com o YARN (*Yet Another Resource Negotiator*), que é um facilitador do modelo de processamento distribuído [125]. O YARN faz monitoramento e alocação dos processadores, disco e memória, para cada execução de progama no *cluster* Hadoop.

Uma típica aplicação Hadoop não faz acesso direto aos recursos computacionais do *cluster*, como visto na Figura 2.3. As aplicações são escritas com um *framework* suportado pelo YARN, que pode ser MapReduce [11], Apache Pig [18], Apache Hive [10], Apache Spark [12], Apache Tez [13] etc. Este *framework*, que está na camada de aplica-

ção, é responsável pelo acesso ao YARN, que por sua vez tem interface com os recursos computacionais e o sistema de arquivos.

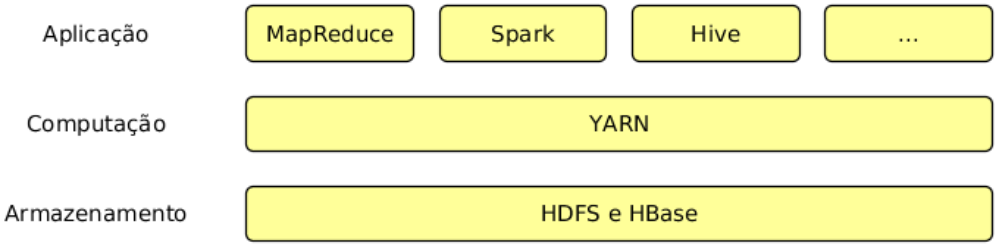

Figura 2.3: Aplicações com YARN. Adaptado de [125].

O processo de submissão de aplicações no YARN envolve dois componentes principais: RM (*Resource Manager*) e NM (*Node Manager*). O RM, geralmente, é executado no mesmo servidor que o *namenode*, e o NM geralmente executa no mesmo servidor que o *datanode* para otimizar os recursos de rede do *cluster*. O RM gerencia os recursos computacionais do *cluster*, e o NM gerencia disco, CPU, rede e memória em cada servidor [14].

2.4.4 Hadoop Distributed File System (HDFS)

O HDFS [9], acrônimo de *Hadoop Distributed File System*, é um sistema de arquivos distribuído para armazenamento de dados muito grandes, da ordem de *gigabytes*, *terabytes* e até mesmo *petabytes*. Ele foi projetado para suportar alta tolerância a falhas executando em máquinas de baixo custo. O HDFS considera que as falhas de hardware são inevitáveis e, por isso, replica partes dos arquivos em vários servidores diferentes para garantir a redundância dos arquivos. Quando algum desses servidores de dados falha, o HDFS recupera automaticamente uma cópia dos arquivos em um servidor disponível que recebeu uma das réplicas daqueles dados [9].

As aplicações que usam HDFS acessam os dados através de *streaming* [9], que, segundo White [125], é o padrão mais adequado para aplicações *big data* com processamento *batch*. Este padrão usa o conceito do *write-once, read many times*, no qual o arquivo é escrito uma única vez e lido várias vezes com alta taxa de transferência. Assim, no HDFS um arquivo pode ser criado, escrito e fechado, contudo, não pode ter seu conteúdo alterado, senão pelas operações de *append* ou *truncate*, e apenas no final do arquivo, ou seja, alterações no meio dos arquivos não são permitidas.

O Hadoop considera que é mais eficiente mover o processamento que mover os dados, considerando a escala de volume dos *datasets* em aplicações *big data* [9]. Com isso, em vez

de mover *gigabytes* ou *terabytes* de dados através da rede em um *datacenter*, o HDFS tem mecanismos para mover as aplicações para o servidor no qual estão localizados os dados, uma estratégia que diminui o tráfego na rede. O HDFS oferece também uma camada de abstração entre o hardware/software e os dados armazenados, o que garante portabilidade entre ambientes heterogêneos.

2.5 Banco de dados NoSQL

O novo paradigma de banco de dados, que não segue o modelo relacional, é geralmente chamado de *Not Only SQL* (NoSQL). Em alguns bancos NoSQL os dados são armazenados em sua forma bruta e a preparação desses dados é feita durante a operação de leitura, uma funcionalidade chamada de *schema-on-read* [27].

Um banco NoSQL pode ser classificado como [61]: (i) orientado a colunas, no qual os valores dos atributos são armazenados em colunas; (ii) orientado a documento, no qual são armazenadas coleções de dados em formatos complexos como JSON e XML; (iii) baseado em grafos, no qual os dados são representados por grafos com vértices, arestas e propriedades; e (iv) chave-valor, no qual os dados são armazenados em mapas associando o valor a uma determinada chave.

Dentre suas características, deve-se citar que um banco NoSQL é flexível, tem API simples, consistência eventual, suporta dados em larga escala e é facilmente replicável [31]. Essas são características gerais, contudo, os bancos NoSQL não seguem as mesmas regras e padronizações que um SGBDR, por exemplo, a maioria dos NoSQL não suporta o SQL, sendo que este suporte pode ser fornecido por *plugins* de terceiros [5, 8, 10].

Outros aspectos do NoSQL são a escalabilidade e a habilidade de manipular quantidades crescentes de dados de forma apropriada [61]. Sistemas de armazenamento de dados escalável e distribuído fazem parte do núcleo da infraestrutura de computação em nuvem e, neste sentido, o NoSQL é o mecanismo ideal para o armazenamento e a recuperação de dados distribuídos em larga escala. As técnicas para uso de bancos NoSQL são detalhadas na Seção 4.4.

2.6 Considerações Finais

Neste capítulo foram apresentados os conceitos sobre as áreas de computação em nuvem e de *big data*. Na área de computação em nuvem, foram definidos os modelos, os tipos de serviço, os atores, os benefícios e os desafios enfrentados para a implantação. Na área de *big data*, além dos conceitos, foi exibida uma classificação para ajudar a explicar as suas

propriedades, as fases dos sistemas de *big data* e, por fim, foi apresentado o ecossistema do Apache Hadoop.

Capítulo 3

Arquiteturas para Sistemas de *Big Data*

Neste capítulo são apresentadas arquiteturas para sistemas de *big data*, detalhando os componentes e as suas interações, bem como tecnologias utilizadas para implantação das soluções. Essas arquiteturas mostram uma visão de alto nível da estrutura necessária para implantar grandes sistemas, principalmente, em ambientes de nuvem computacional.

3.1 Trabalhos Relacionados

A pesquisa de arquiteturas e *frameworks* para *big data* está se desenvolvendo de acordo com as mudanças na tecnologia e nas demandas de novas soluções para problemas que lidam com uma quantidade crescente de dados. Para tanto, novas alternativas são propostas utilizando processamento distribuído, paralelo e escalável, que serão detalhadas no decorrer deste capítulo. Estes trabalhos estão organizados nas seguintes categorias:

- Arquiteturas para *Big Data*: arquiteturas usadas para criar sistemas de *big data* em domínios variados como mídias sociais e proteção contra desastres naturais;
- Arquiteturas para Análise de Dados: arquiteturas para criação de plataformas para análises *ad hoc* de dados;
- Infraestrutura para *Big Data*: trabalhos relacionados com soluções para monitoramento, virtualização, implantação, provisionamento, performance de sistemas de *big data*;
- Arquiteturas de Referência: arquiteturas que descrevem de forma teórica a estrutura e as técnicas necessárias para criação de soluções de *big data*.

3.1.1 Arquiteturas para *Big Data*

O trabalho de Gao [55] propõe uma nova arquitetura escalável e integrada para processamento paralelo de dados textuais gerados em grande velocidade nas mídias sociais, além de comparar a performance de diversas ferramentas e algoritmos. A arquitetura, chamada de *Cloud* DIKW, é composta por módulos que combinam soluções para armazenamento e processamento de *big data*. Para atender a essa nova demanda gerada pelas mídias sociais, a arquitetura suporta consultas em paralelo, análise com processamento *batch* e análise com processamento em tempo real.

Como visto na Figura 3.1, a arquitetura de [55] é composta por três elementos: (1) núcleo de armazenamento, composto por banco de dados NoSQL (detalhado na Seção 2.5) e por um *framework* personalizado para indexação dos textos; (2) módulo para análise *batch*, composto por *frameworks* para processamento em paralelo; e módulo para análise de dados em *streaming*, composto por *frameworks* para processamento paralelo de *streaming* [55]. A escolha de ferramentas mostra-se adequada para a solução, assim como a personalização do HBase e do Storm para melhorar a performance dos dados textuais.

A arquitetura de [55] mostra uma solução escalável para a análise de mídias sociais usando tecnologias *big data*, entretanto, está limitada à indexação textual e não é possível estendê-la para outros domínios, como os sistemas governamentais compostos por fontes de dados diversas. No que diz respeito à elasticidade, não está claro como pode ser resolvida a questão da carga de trabalho variável, ou seja, como seria feito o provisionamento dos recursos para atender a picos momentâneos de processamento, uma vez que o *dataset* utilizado era de tamanho conhecido. Esta é a lacuna que a arquitetura proposta pretende ocupar adicionando suporte para dados governamentais e provisionamento automatizado de recursos.

O estudo de Zheng *et al.* [131] apresenta definições para os serviços relacionados com a área de *big data*. Segundo os autores, geralmente, o BDaaS é dividido em três camadas de abstração, compostas pelo (i) *Big Data Infrastructure as a Service*, que se baseia no IaaS para endereçar demandas de processamento massivo de dados; (ii) *Big Data Platform as a Service*, que é baseado no PaaS e no qual estão incluídos os serviços de armazenamento, acesso a dados e bancos de dados, acessados por meio de APIs Web disponíveis em interfaces REST; e o (iii) *Big Data Analytics Software as a Service*, que é baseado no SaaS e no qual são disponibilizados softwares para análise de dados, geralmente, usando o ecossistema do Hadoop por meio de interfaces Web para execução de consultas e *scripts*.

Neste trabalho [131] dois modelos merecem atenção. O primeiro é o *Data as a Service* (DaaS), que é a habilidade de definir listas de dados em um serviço de nuvem, e permitir acesso controlado por meio de uma API de *Web Services* RESTful. Esta API permite apenas operações de consultas básicas, uma vez que não é possível utilizar SQL. Este

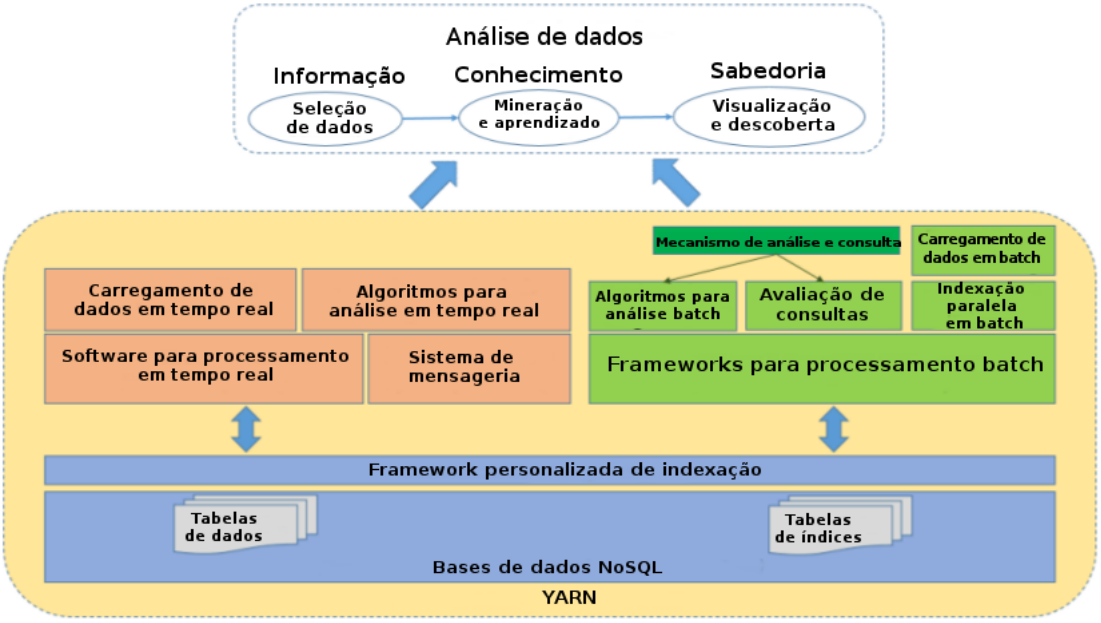

Figura 3.1: Arquitetura para análise de dados de mídias sociais. Adaptado de [55].

modelo é adequado para provedores de dados públicos. O segundo é o *Database as a Service* (DBaaS), um modelo que oferece o serviço de banco de dados por meio da nuvem, que pode ser acessado através de APIs. Os serviços podem ser fornecidos por SGBDRs, NoSQL e bancos de dados *in-memory*.

As visões apresentadas em [131] mostram uma classificação de BDaaS que está parcialmente de acordo com a proposta nesta dissertação. Contudo, há diferenças na forma como é definido o armazenamento, uma vez que [131] acredita que o *cloud storage* e o *Storage as a Service* (StaaS) são recursos distintos. Outro ponto é que o artigo destaca o uso do Hadoop, uma vez que este não é o único software usado nas soluções citadas.

A Arquitetura Lambda, proposta por Marz e Warren [89], foi projetada com base nos princípios de escalabilidade, simplicidade, imutabilidade dos dados, *frameworks* de processamento em *batch* e em tempo real. A arquitetura foi criada a partir da observação dos problemas apresentados pelos sistemas de informação tradicionais, como a complexidade da operação, a adição de novas funcionalidades, a recuperação de erros humanos e a otimização de performance. Segundo os autores [89], a Arquitetura Lambda é genérica e pode ser usada em qualquer sistema de informação. Esta arquitetura usa técnicas e ferramentas de *big data* para processar e armazenar os dados, incluindo tecnologias para o processamento em *batch*, o banco de dados NoSQL para gerenciamento de dados e o sistema de mensageria para ingestão de dados.

Novamente citando [89], o objetivo é combinar os benefícios de cada tecnologia para minimizar os pontos fracos. Por exemplo: o Hadoop é eficiente para processamento

paralelo, mas não é eficiente nas consultas a registros individuais. Assim, para resolver essa deficiência do Hadoop, a arquitetura propõe usar um banco NoSQL de acesso rápido. Para organizar os elementos internos, a arquitetura é dividida em três camadas, que podem ser vistas na Figura 3.2, as quais são:

- Camada de lote (*Batch layer*): armazena a cópia principal dos dados e pré-processa as visões em lote com o sistema de processamento em *batch* (Hadoop);
- Camada de serviço (*Serving layer*): armazena o resultado do processamento em *batch* em um sistema de gerenciamento de dados para consultas, como um banco NoSQL;
- Camada de velocidade (*Speed layer*): processa os dados novos que chegam enquanto ocorre o processamento em *batch*, garantindo a execução da consulta com dados em tempo real.

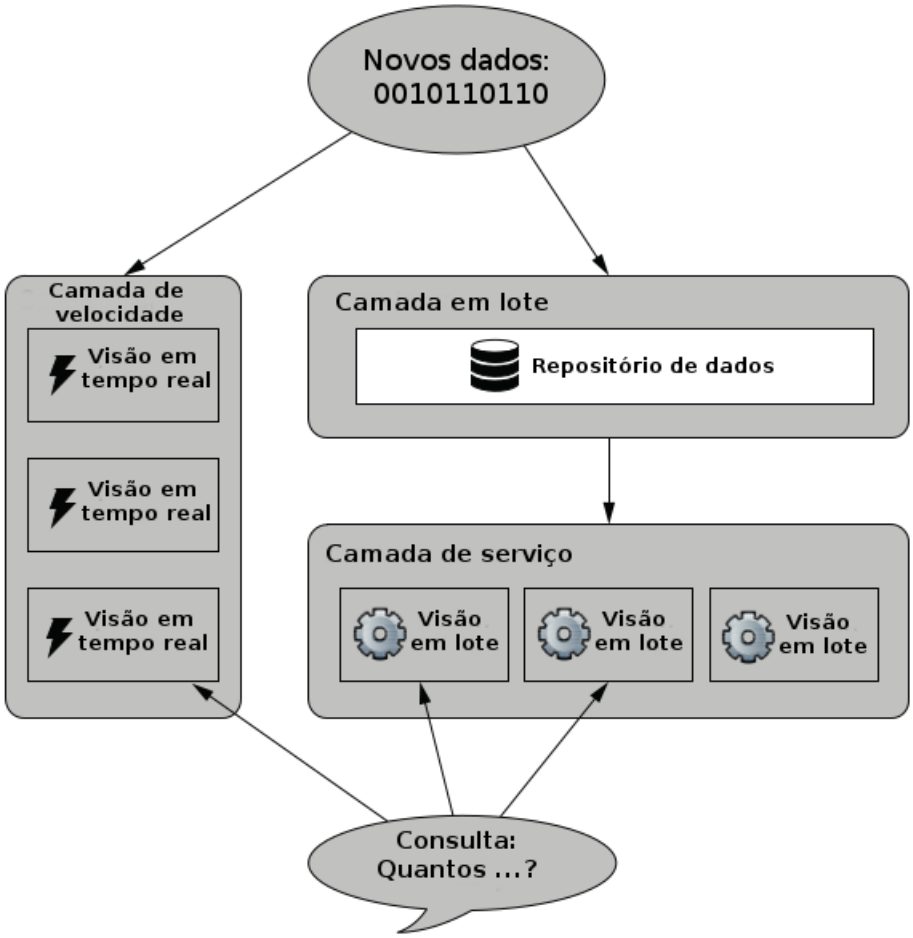

Figura 3.2: Diagrama da Arquitetura Lambda. Adaptado de [89].

A solução proposta em [89] tem se popularizado na área de *big data* por ser flexível, o que permite que ela atenda a cenários diferentes. Desta forma, é uma arquitetura eficiente para a criação de soluções de processamento massivo. O trabalho explica em detalhes como criar um sistema usando a Arquitetura Lambda, entretanto, o procedimento para implantação em nuvem não está definido, mesmo quando o modelo IaaS é citado como opção para infraestrutura, como é o caso do AWS. Neste sentido, a presente dissertação apresenta uma estratégia para implantação da solução em nuvem.

3.1.2 Arquiteturas para Análise de Dados

O trabalho de Jambi [68] diz que existe uma grande demanda para técnicas e ferramentas para processamento de dados em *streaming*. Neste estudo, o autor utiliza dados do Twitter[1] com o objetivo de prever eventos de crises, analisando o comportamento social durante desastres. Pelo tipo dos dados e a urgência na resposta, as consultas precisam ser executadas em tempo real, o que demanda uma arquitetura de software robusta, confiável e adaptável, uma vez que os usuários precisam executar consultas *ad hoc*. Em um cenário de desastres da natureza, como por exemplo terremotos, o principal benefício da plataforma é oferecer respostas em tempo real.

Os requisitos do projeto [68] envolvem a análise de *big data* em tempo real por meio de um mecanismo de consulta flexível, que disponha de respostas rápidas e de alta disponibilidade. Assim, o resultado desta pesquisa apresenta uma nova plataforma para análise de *big data* por meio de serviços REST, que pode ser vista na Figura 3.3. Para avaliar os resultados, os autores criaram um protótipo chamado EPIC *Real-Time* que combina o processamento *streaming* e *batch*, e usa como base a Arquitetura Lambda [89]. O protótipo utiliza *datasets* específicos, como mídias sociais, contudo, é possível estender a arquitetura para outros domínios com poucas linhas de código, uma vez que a solução apresenta camadas de abstração. Segundo o autor, as implementações podem ser feitas com menos de 40 linhas de código. Contudo, a escalabilidade do protótipo é obtida com a instalação manual de novos servidores, uma atividade onerosa que ocorreu porque não foi utilizado um ambiente de computação em nuvem, que é uma pendência endereçada por esta dissertação.

Em Simmhan *et al.* [116] é apresentada uma plataforma de nuvem para análise de dados e a implementação de um sistema de *big data* para otimização do uso de recursos hídricos e elétricos, uma operação chamada de *Dynamic Demand Response* (D^2R). A aplicação construída pode detectar uma desproporção entre o suprimento e a demanda de recursos, e então pode realizar as devidas correções. Os dados são coletados por meio de sensores e fontes de dados dinâmicas, e analisados em tempo real, incluindo análises com modelos

[1] https://twitter.com/

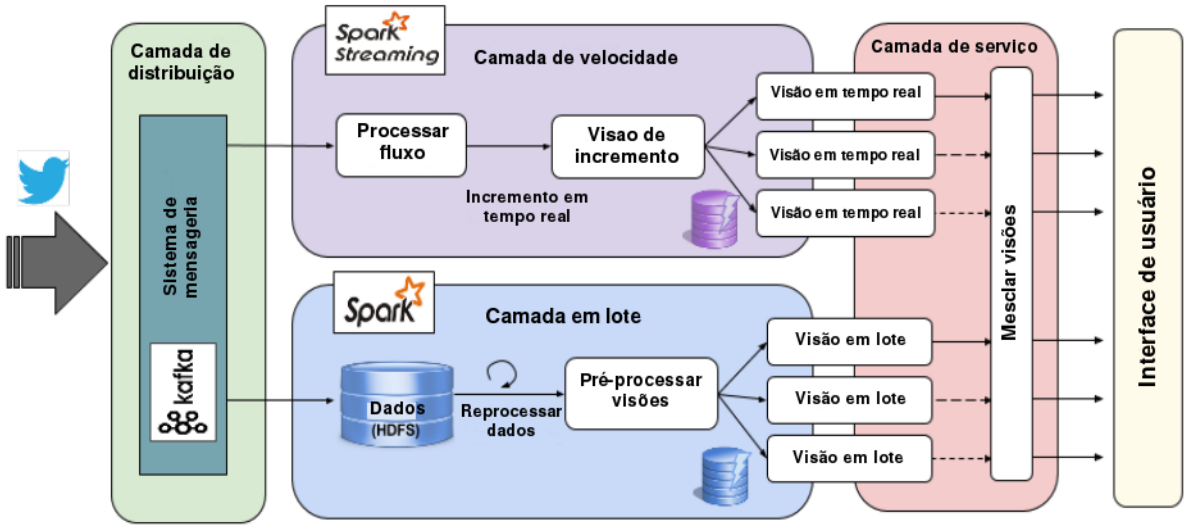

Figura 3.3: Arquitetura usada no EPIC *Real-Time*. Adaptado de [68].

de *machine learning* que são usados para prever a demanda futura. Dado esse cenário complexo, a solução de tecnologia selecionada foi o Hadoop, executado em infraestrutura de nuvem privada. Os autores consideram que a escalabilidade e a capacidade de resposta são fatores chave para a operacionalização da plataforma e, por fim, verificaram o valor da computação em nuvem associada à análise de dados para gerenciamento inteligente e sustentável de sistemas físicos. A solução apresentada em [116] usa os modelos IaaS e PaaS em nuvens públicas e privadas para análises de dados, e não é uma arquitetura de uso geral extensível a outros domínios.

O *framework* proposto em Eftekhari *et al.* [46] é um SaaS para consultas *ad hoc* em diferentes fontes de dados, incluindo SGBDR e armazenamento de *big data*, implantado em nuvem privada. Os dados das fontes são consolidados em um armazenamento *big data* que funciona como o ponto central para a integração e a análise de dados. A arquitetura desenvolvida é extensível e permite a utilização de outras soluções de armazenamento e de análise de dados, por exemplo Apache HBase [8] ou linguagem R [112]. Essa flexibilidade é possível porque as conexões entre as camadas da arquitetura são fracamente acopladas, de forma que as atualizações de software em pontos específicos não impactam toda a solução. A validação é feita por meio de uma prova de conceito, na qual é entregue uma interface de usuário simplificada para consulta aos dados sem necessidade de conhecimento de programação.

Neste contexto, a Figura 3.4 mostra a proposta de arquitetura Eftekhari *et al.* [46], na qual são identificadas três camadas: interface de usuário, aplicação e recursos. O usuário

final interage por meio da camada de interface, que aciona a camada de aplicação, a qual contém os adaptadores apropriados de acordo com a demanda. Assim, estão disponíveis adaptadores para ETL, SGBDR e *big data*. A última camada é responsável pelos recursos de armazenamento e de *big data*. As ferramentas usadas no protótipo foram o MySQL [97] e Hadoop [6].

O trabalho de [46] apresenta ideias em comum com a presente dissertação, como o uso de ponto central para integração de dados e nuvem privada. Contudo, usa ferramentas diferentes e, com isso, não é possível separar o *cluster* dos dados, o que gera acoplamento, uma característica que se pretende evitar em sistemas de *big data* modernos. Assim, esta dissertação propõe que dados e *cluster* sejam independentes com o uso de *data lakes* e *object stores*.

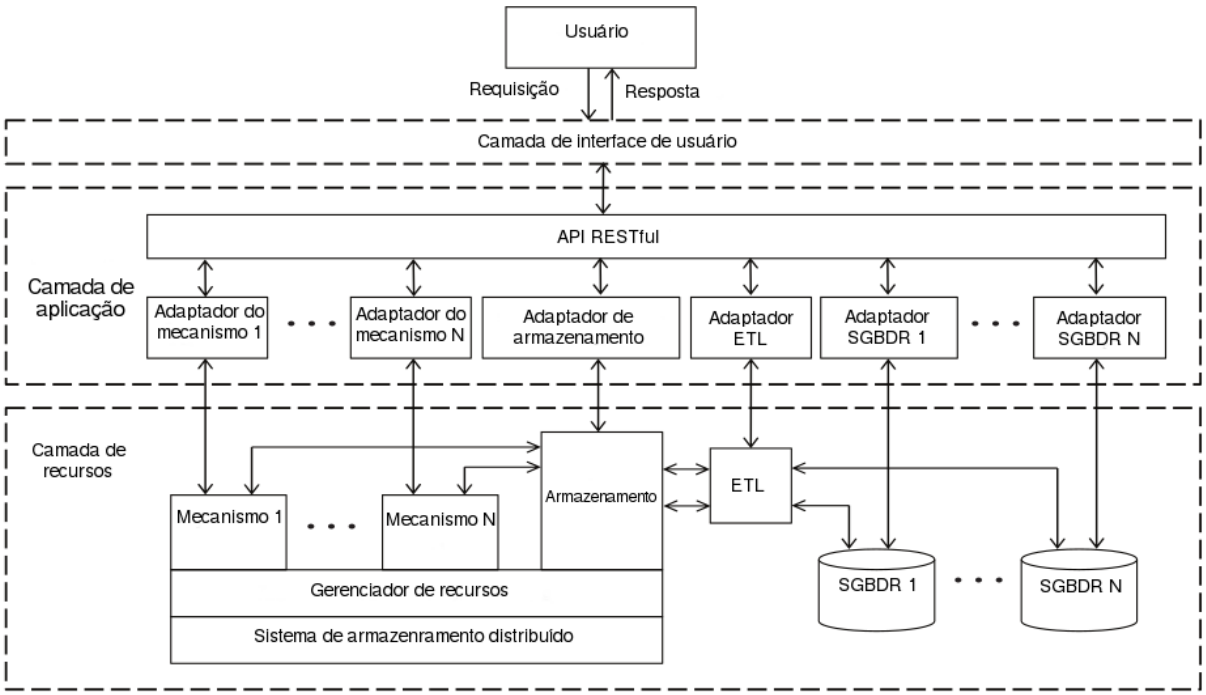

Figura 3.4: Arquitetura proposta para consultas *ad hoc*. Adaptado de [46].

3.1.3 Infraestrutura para *Big Data*

Em Horey *et al.* [63] os autores propõem uma solução para o problema da complexidade da implantação de nuvem privada por meio de uma camada de abstração com máquinas virtuais, de forma que os usuários não precisem lidar com as máquinas individuais ou com sua organização interna. Neste contexto, é apresentado o Cloud-Get, que é um gerenciador de pacotes que permite a instalação de serviços distribuídos em um ambiente

de computação em nuvem, inclusive soluções de *big data*. Como resultado, os usuários podem instanciar e modificar serviços distribuídos usando comandos simples.

A ferramenta desenvolvida em [63] facilita a instalação de plataformas de *big data*, incluindo softwares como o Hadoop e o HBase [8], bem como sua configuração. A solução foi desenvolvida no CloudStack e os autores colocaram como trabalho futuro uma versão para OpenStack e AWS. É um projeto inicial que apresenta funcionalidades básicas, limitadas ao provisionamento do *cluster*, contudo, não contempla itens como o monitoramento, que é importante para sistemas corporativos.

Em Thaha *et al.* [120] os autores exploram a utilização do Hadoop virtualizado, sendo executado no OpenStack [104], e propõem uma estratégia de provisionamento de *cluster* baseada na localização dos dados que utiliza armazenamento anexado por meio de *Storage Area Network* (SAN) ou *Network-Attached Storage* (NAS). A localização dos dados é uma questão importante na área de processamento distribuído porque a latência nas operações de leitura e de escrita, entre os servidores, resulta em aumento no tempo de execução das aplicações. Neste estudo, os autores calculam o impacto da localização dos dados em *clusters* Hadoop considerando armazenamento (i) efêmero, aquele com acesso direto nativo no sistema de arquivos; (ii) em bloco, por meio de um dispositivo de armazenamento particionado e montado no servidor; e (iii) objeto por meio da API REST.

A conclusão do estudo de [120] é que a criação dos *clusters* deve ser feita o mais próximo possível dos dados, se possível, na mesma máquina. A virtualização e a localização dos dados são características do provisionamento automatizado do OpenStack Sahara [101], assim, é possível configurar o *cluster* para usar VMs próximas dos dados, minimizando o uso de rede. Entretanto, como visto no Capítulo 4, a conclusão desta dissertação segue em outra direção, com o uso de *object stores*.

O trabalho de Chang [25] apresenta um sistema de *disaster recover* (DR) para *big data* em nuvem privada. No estudo, o autor afirma que a segurança é um desafio na adoção de soluções *big data*, e que o gerenciamento de dados na nuvem deve incluir um *backup* ou plano de contingência para minimizar os impactos em caso de desastre. A proposta de solução é usar várias técnicas, garantindo que um ou mais métodos possam recuperar e restaurar dados rapidamente, mesmo para *backups* com vários *terabytes*. Para isso são aplicados métodos que usam o TCP/IP, recuperação de *snapshots* e uma solução híbrida que aplica os dois métodos juntos. Como resultado, a solução consegue recuperar um *backup* de 1 TB de dados em um intervalo de tempo que varia entre 10 a 20 minutos, uma performance aceitável nos ambientes corporativos.

Xu *et al.* [129] apresentam os resultados de uma pesquisa que compara a implantação de *clusters* Hadoop em nuvem privada e em máquinas físicas, listando as vantagens e as desvantagens de cada estratégia. A principal consideração é sobre o I/O de disco e

rede, que tem performance diferente entre máquinas físicas e virtualizadas. Mesmo nas máquinas virtualizadas há diferença de performance quando as aplicações estão rodando no mesmo disco físico ou em discos separados. Para melhorar a eficiência do processamento no Hadoop foram usados algoritmos de escalonamento, com o LATE (*Longest Approximate Time to End*). A diferença de performance entre máquinas físicas e virtuais era esperada, contudo, as vantagens da virtualização superam as perdas de desempenho, uma vez que o gerenciamento dos serviços pode ser mais caro que a aquisição dos equipamentos na área de *big data*.

O estudo de Adnan *et al.* [2] incentiva as empresas a criarem sua infraestrutura de *big data* usando Hadoop na nuvem privada, pública ou híbrida, com o objetivo de minimizar os problemas que uma implantação *big data* oferece tradicionalmente. A abordagem proposta é composta por uma (a) nuvem de controle, que recebe as requisições dos usuários; (b) várias nuvem secundárias, que usam infraestrutura privada ou pública e que devem ser baseadas em Hadoop para melhor performance; (c) tabela de nuvens registradas, com a lista das nuvens disponíveis na infraestrutura e sua prioridade para responder às requisições da nuvem de controle. Com essa arquitetura, os resultados mostraram aumento na performance e na capacidade de atender a requisições. Entretanto, deve-se observar que os experimentos foram feitos com sistema operacional Windows e arquivos de 100 MB, um ambiente insuficiente para avaliar a performance de sistemas de *big data*.

A performance do Hadoop em nuvem IaaS é analisada no trabalho de Conejero *et al.* [34], no qual é reportado que aplicações baseadas em Hadoop precisam de muitos recursos computacionais para realizarem análises em *big data*, e que a integração do Hadoop com a computação em nuvem não é trivial. Um *cluster* Hadoop demanda múltiplas VMs, que devem ser integradas com o ambiente físico para garantir a execução sob demanda das aplicações. Neste cenário, os autores investigam o impacto da execução de múltiplas instâncias concorrentes do Hadoop sob o aspecto da eficiência na alocação de recursos na nuvem.

Ainda em [34] são discutidas as estratégias de implantação (i) horizontal, na qual o *cluster* é alocado em VMs na mesma máquina física para economizar energia; (ii) vertical, na qual a implantação do *cluster* usa VMs espalhadas em todos os nós físicos disponíveis; (iii) *master-apart*, que é uma variação da estratégia horizontal, na qual a VM do *master* fica em uma máquina física, enquanto as VMs dos *workers* ficam consolidadas em outro servidor físico; e (iv) *complete-spread*, na qual todas as VMs são espalhadas entre as máquinas físicas disponíveis, como na estratégia vertical, mas com a limitação de que cada máquina física tem apenas um *master*. O principal resultado da pesquisa é que a estratégia de implantação tem impacto significativo na performance apenas quando há poucos *clusters* em execução. Dessa forma, os autores concluem que a consolidação do

cluster na mesma máquina física tem impactos positivos no consumo de energia.

Do ponto de vista de ferramentas, a pesquisa de Vogel *et al.* [123] analisou soluções de IaaS para implantação de nuvem privada, observando questões como flexibilidade, performance e resiliência. As ferramentas estudadas foram o OpenNebula [100], OpenStack [104] e CloudStack [33], que foram comparadas entre si e também com o hardware nativo sem ferramenta de IaaS. A conclusão do trabalho discute e compara as ferramentas de implantação de nuvem privada, na qual não se verificam significativas diferenças entre as ferramentas, e que o *overhead* da virtualização também não é expressivo.

Em Corradi *et al.* [39], os autores propõem um mecanismo dinâmico para alocação de recursos, de forma a garantir elasticidade e eficiência ao módulo de gerenciamento de *clusters* Hadoop do OpenStack, chamado de Sahara. Este novo mecanismo foi chamado de *Elastic Sahara MApReduce* (ESAMAR). O ESAMAR monitora a performance do *cluster* Hadoop para dimensionar a quantidade adequada de recursos para a carga de processamento. Nesse estudo foi desenhado um modelo de gerenciamento e de provisionamento que opera em conjunto com técnicas de monitoramento em nível de sistema, que automaticamente redimensiona o tamanho do *cluster*, otimizando o balanceamento de carga. O protótipo deste sistema foi testado em situações de carga real e, então, foi disponibilizado para a comunidade *open source* do Sahara.

3.1.4 Arquiteturas de Referência

O trabalho de Maier [86] apresenta uma arquitetura de referência para *big data*, na qual os autores partem do pressuposto de que faltam orientações e abordagens claras para a sua efetiva adoção. O estudo organiza as tecnologias e os componentes funcionais, criando uma base para implantação de soluções de *big data*. Assim, o foco desse estudo [86] é a questão do software disponível para implantação de soluções de *big data*, e os autores não fazem considerações sobre o provisionamento dos recursos computacionais, como o projeto do hardware do *cluster*.

Em Hu *et al.* [65] é apresentada uma extensa revisão bibliográfica da área de *big data*, um tutorial para plataformas de análise de *big data* e uma arquitetura para sistemas de *big data* que é dividida em três camadas: (i) camada de aplicação, (ii) camada de computação e (iii) camada de infraestrutura. A camada de aplicação, como visto na Figura 3.5, é composta por um mecanismo de consulta, algoritmos de *clustering*, classificação e recomendação. A camada de computação inclui diversos componentes, incluindo os *frameworks* para *big data*, NoSQL/SQL, sistema de arquivos e integração. A última camada gerencia a infraestrutura de servidores, rede e armazenamento. Esta arquitetura mostra seus elementos em nível bastante conceitual, na forma de uma revisão da literatura na área, citando inclusive arquiteturas de grandes empresas como Walmart, Amazon e

Facebook. Entretanto, o estudo de Hu *et al.* [65] não apresenta um protótipo, limitando-se ao escopo teórico.

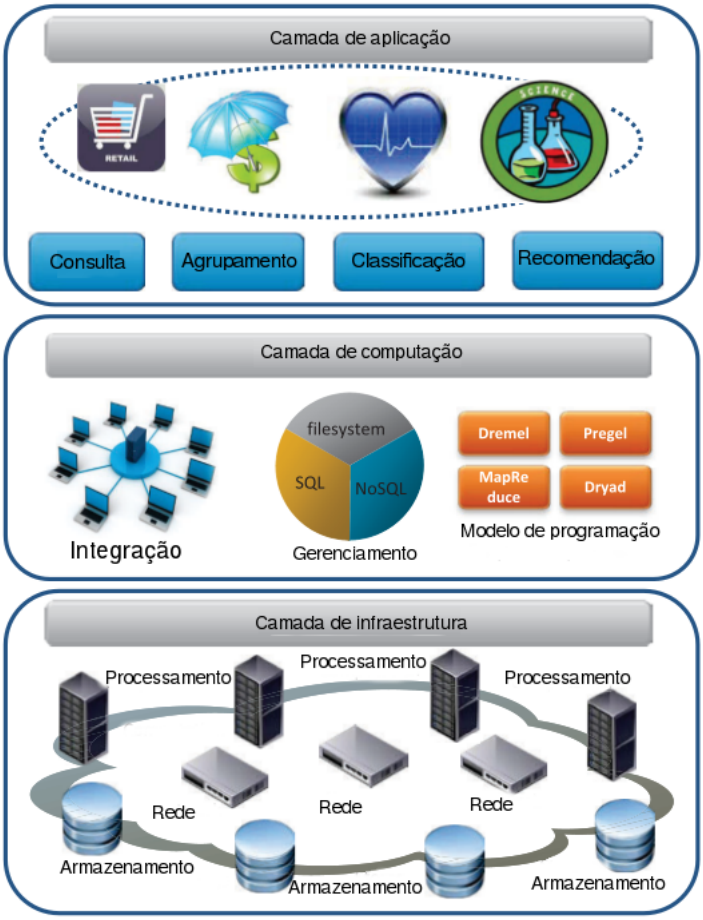

Figura 3.5: Arquitetura em camadas para sistemas de *big data*. Adaptado de [65].

Em Chang [26] os autores mostram uma arquitetura de referência para soluções de *big data* que pode ser vista na Figura 3.6. O objetivo é criar um modelo conceitual de arquitetura para arquitetura *big data*, sem referência a tecnologias ou ferramentas específicas. Os dois eixos mostrados representam a Informação (horizontal) e a Tecnologia da Informação (vertical). No eixo da informação, o valor é criado pelas tarefas de coleção de dados, integração e análise. No eixo da TI, o valor é criado pelo provisionamento de rede, infraestrutura, plataformas e aplicações, bem como outros serviços de suporte. Neste modelo são definidos os seguintes componentes lógicos funcionais:

- Orquestrador do sistema: define e integra as atividades em um sistema operacional vertical. O orquestrador pode ser uma pessoa, um software ou uma combinação dos dois, e é responsável por configurar e gerenciar os outros componentes da arquitetura para implementar a carga de trabalho através de uma interface gráfica, ou atribuindo diretamente a carga de trabalho ao recurso computacional;

- Provedor de dados: inclui novos dados ou fontes de informação no sistema;
- Provedor de aplicação *big data*: encapsula a lógica de negócio e a funcionalidade para ser executada pela arquitetura. Inclui atividades como a coleção, a preparação, a análise, a visualização e o acesso aos dados;
- Provedor de *framework big data*: consiste de uma ou mais tecnologias para garantir flexibilidade e atender ao requisitos que são definidos pelo provedor de aplicação *big data*. É o componente que recebe mais atenção da indústria e o que tem mais informação disponível. Este componente é discutido em detalhe na Seção 2.4;
- Consumidor dos dados: são os usuários finais e outros sistemas que usam o resultado produzido pelo provedor de aplicação *big data*.

Muitos dos avanços na área de *big data* se refletem no componente provedor de *frameworks*, que atende a um amplo leque de funcionalidades. Dessa forma, para atender aos diversos requisitos das demandas levantadas pelo provedor de aplicação *big data*, uma típica implementação *big data* conta com múltiplos *frameworks*. Assim, na Figura 3.6 é possível verificar que o componente provedor de *framework big data* é composto de três subcomponentes [26]:

- *Frameworks* de infraestrutura: suportam recursos de rede, computação, armazenamento e recursos físicos (energia, resfriamento, segurança etc);
- *Frameworks* para plataforma de dados: responsáveis pela organização lógica de dados. Aplica-se a situações variadas, desde arquivos de texto delimitados até *data stores* distribuídas. O método de acesso ao dados inclui APIs para consulta ou *Structured Query Language* (SQL);
- *Frameworks* para processamento: definem como o processamento dos dados é organizado e executado. Eles são tipicamente focados na manipulação de dados e podem ser orientados a processamento *batch* ou *streaming*.

A arquitetura de referência de Chang [26] é abstrata, ou seja, não tem um protótipo para avaliação da performance e das tecnologias, limitando-se a fazer o detalhamento teórico dos elementos deste tipo de arquitetura. Ela deve ser usada para nortear a construção de novas arquiteturas de sistemas de *big data*.

Em Bhagattjee [23], o autor define que BDaaS é um *framework* para computação distribuída, horizontalmente escalável e baseado em nuvem, projetado para manipular grandes *datasets* (*big data*). Entretanto, devido à quantidade de tecnologias disponíveis, é difícil identificar as soluções adequadas para cada demanda. Como resultado, o desenvolvimento de sistemas de *big data* acaba por envolver altos custos tanto para gerenciamento

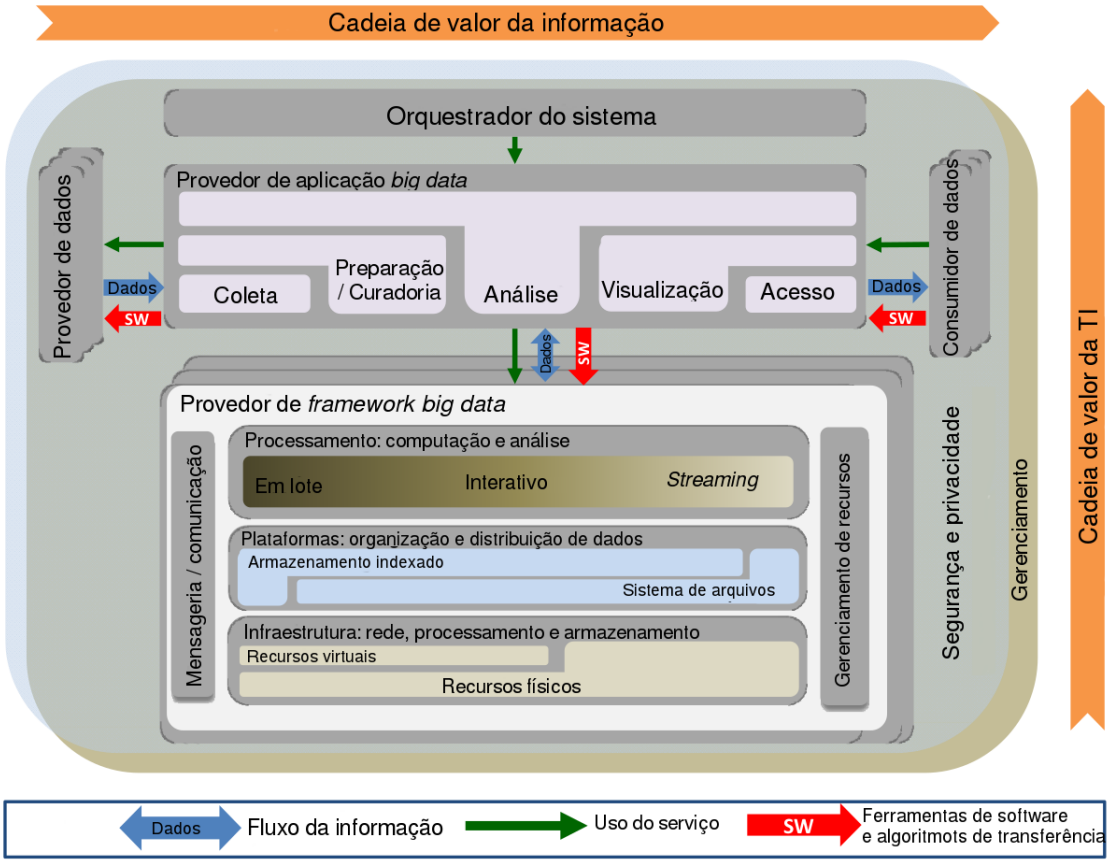

Figura 3.6: Arquitetura de referência para *big data*. Adaptado de [26].

da infraestrutura quanto para obtenção de mão de obra especializada. Como proposta de solução, Bhagattjee [23] apresenta um *framework* para ajudar usuários e fornecedores de tecnologia a identificar e classificar tecnologias de *big data* baseadas em plataforma de nuvem. O *framework* é descrito em camadas, sendo cada uma com um conjunto de responsabilidades e as ferramentas adequadas para implementação.

Neste contexto, Assunção *et al.* [21] concluem que o processamento *big data*, usando computação em nuvem dispõe de várias soluções no mercado, e essa quantidade de ferramentas é motivado pela complexidade inerente à análise de dados. Para selecionar corretamente as ferramentas e realizar as análises, é necessário mão de obra especializada. Neste cenário, é reportado que os modelos de BDaaS e de *Analytics as a Service* (AaaS) estão se tornando populares. Contudo, sua adoção ainda tem pontos em aberto. Por exemplo, este tipo de solução ainda demanda contratos mais bem definidos, uma vez que é difícil medir a qualidade e a confiabilidade dos dados e dos resultados, previsibilidade do tempo de execução, garantias nos métodos e disponibilidade de especialistas. Com isso, há uma lacuna entre as funcionalidades das ferramentas e os clientes.

3.1.5 Comparativo

Tendo em vista todas as arquiteturas apresentadas neste capítulo, fica claro que a implantação de *big data* como um serviço na nuvem privada (BDaaS) é uma área que oferece desafios, e demanda novas pesquisas no sentido de diminuir a complexidade da implantação da infraestrutura para o uso efetivo de soluções de *big data* nas organizações.

Assim, um *framework* BDaaS pode ajudar as organizações a se moverem rapidamente do ponto inicial, que é a pesquisa de tecnologias *big data*, para a fase final de implantação da solução. Mesmo as fases que envolvem projetos-piloto seriam beneficiadas com o uso de tecnologias baseadas em nuvem, resultando na diminuição do tempo necessário para desenvolver a solução e, consequentemente, os custos do sistema.

Diante do exposto, nota-se que empresas e centros de pesquisa têm demonstrado interesse em soluções para implantação de nuvem privada, com o objetivo de otimizar a performance, o contingenciamento, a segurança e a disponibilidade dos recursos do *datacenter*. Da mesma forma, a área de *big data* desperta atenção como uma forma de expandir a capacidade de processamento do *datacenter* tradicional.

A nuvem privada e o *big data* são boas alternativas para cargas de trabalho intensas, não apenas aplicações científicas, mas também aplicações do governo ou de telecomunicações. Contudo, enquanto a computação em nuvem pública é uma área consolidada, amplamente pesquisada e conhecida, a computação em nuvem privada ainda é um campo pouco explorado. Isso acontece porque a integração entre *big data* e computação em nuvem não é trivial, resultando em desafios técnicos. A diferença na evolução entre a nuvem pública e a privada é motivada principalmente pelos altos investimentos em pesquisa e desenvolvimento feitos pelos maiores provedores, como Amazon e Microsoft.

A Tabela 3.1 apresenta um comparativo entre os trabalhos referenciados nesta dissertação, na qual se verifica que as arquiteturas disponíveis não contemplam todas as características desejáveis para a criação de sistemas de *big data* no Governo Federal. Estes atributos serão endereçados na proposta apresentada no Capítulo 4, o que justifica o custo de implantação.

- *Batch*: Processamento em *batch*;
- RT: Processamento em tempo real;
- Hadoop: Suporte para instalação e configuração automatizada de *cluster*;
- NoSQL: Suporte para instalação e configuração automatizada de NoSQL;
- Cloud: Suporte à computação em nuvem (auto-serviço, acesso à rede, agrupamento de recursos, elasticidade e serviço mensurável);

- Private: Suporte para implantação em nuvem privada;
- BDaaS: Suporte para *Big Data as a Service*;
- API: Suporte para APIs;
- PoC: Validação de PoC (*Proof-of-Concept*) com grande volume de dados corporativos;
- *Ad Hoc*: capacidade para executar consultas *ad hoc* no dataset;
- Referência: apresenta uma arquitetura de referência com definições, técnicas, estruturas e as respectivas ligações.

3.2 Considerações Finais

Neste capítulo foram apresentadas as arquiteturas e as tecnologias usadas para criar sistemas de *big data* em contextos acadêmicos e da indústria. Assim, foi possível identificar a tendência de associar sistemas de *big data* com a computação em nuvem para diminuir a complexidade da implantação e agilizar a entrega da solução.

Algumas das características presentes nas arquiteturas estudadas neste capítulo, e que serão utilizadas na formulação da proposta desta dissertação, são a divisão em camadas, a utilização de *frameworks* para processamento em *batch* e em tempo real, o uso de bancos NoSQL e as técnicas específicas para criação dos sistemas de *big data*.

Dado esse contexto, o próximo capítulo apresenta uma arquitetura para implantação de sistemas de *big data*, na forma de serviço, em nuvens privadas, que atende a requisitos de aplicações corporativas, que pode ser estendida para as áreas nas quais a demanda está prevista nos cenários de uso.

Tabela 3.1: Comparativo entre os Trabalhos Relacionados.

	Característica										
Trabalho	*Batch*	RT	Hadoop	NoSQL	Cloud	Private	BDaaS	API	PoC	*Ad Hoc*	Referência
Gao [55]	X	X							X	X	
Zheng *et al.* [131]							X	X	X	X	
Marz e Warren [89]	X	X			X			X	X		
Jambi [68]	X	X	X	X	X	X	X	X	X		
Simmhan *et al.* [116]	X	X	X		X	X			X		
Eftekhari *et al.* [46]	X	X	X	X	X	X	X	X	X		
Horey *et al.* [63]	X	X	X		X						
Thaha *et al.* [120]	X	X	X		X	X		X	X		
Chang [25]	X	X			X	X			X		
Xu *et al.* [129]	X				X	X		X			
Adnan *et al.* [2]	X		X		X	X			X		
Conejero *et al.* [34]	X		X		X	X					
Vogel *et al.* [123]	X		X		X	X			X		
Corradi *et al.* [39]	X		X		X	X			X		
Maier [86]	X	X	X	X	X						X
Hu *et al.* [65]	X	X	X	X	X			X			X
Chang [26]	X	X		X	X	X			X	X	X
Bhagattjee [23]	X	X	X	X	X		X	X		X	X
Assunção *et al.* [21]	X	X	X	X	X	X	X	X		X	X

Capítulo 4

Arquitetura para BDaaS em Nuvem Privada

Neste capítulo será apresentada a arquitetura para sistemas de *big data* em nuvem privada proposta neste trabalho, usando cenários baseados nas demandas dos órgãos do Governo Federal, conforme descrito na Seção 4.5.2. O modelo de serviço escolhido foi o *Big Data as a Service* (BDaaS) para incluir as facilidades da computação em nuvem.

Esta infraestrutura propõe tornar os dados corporativos disponíveis em um ponto centralizado com alta disponibilidade e performance, incorporando as facilidades da computação em nuvem para encurtar o tempo necessário para implantar sistemas de *big data* que operem os mesmos cenários descritos nesta proposta.

Desta forma, são descritos os padrões arquiteturais desde seus conceitos e ideias principais, em um plano funcional, seguindo até as tecnologias de implementação. Assim, a arquitetura proposta contempla tanto as abstrações de alto nível como também as ferramentas de nível mais baixo.

Com isso, as contribuições deste capítulo da dissertação são: (i) cenários de uso dos sistemas de *big data*, (ii) agrupamento das funcionalidades relacionadas em módulos da arquitetura, (iii) mapeamento das funcionalidades do sistema de *big data* em tecnologias de implementação, (iv) prova de conceito que usa a arquitetura para criar um sistema de *big data* e (v) técnicas para a criação de sistemas de *big data* em nuvem privada.

4.1 Motivação para *Big Data*

As tecnologias de *big data* estão presentes em diversas áreas da sociedade, incluindo indústria, economia, saúde, ciência, segurança e administração pública, sendo esta última o foco desta dissertação. Entretanto, é importante dizer que as soluções de *big data* não são as únicas alternativas para processamento e análise de quantidades massivas de

dados. Assim, é importante caracterizar a necessidade da adoção de *big data* e da própria computação em nuvem.

A computação em nuvem e a virtualização são amplamente utilizadas, enquanto que a pesquisa e a adoção de *big data* na nuvem ainda está em estágios iniciais [61], de forma que o primeiro ponto a ser observado é que o rápido crescimento na quantidade de dados gerados nestas organizações limita, em determinados casos, a capacidade das tecnologias tradicionais de armazenar e processar as informações corporativas. Os desafios começam com a coleta dos dados em sua origem e vão até sua visualização, passando pelo custo de aquisição dos equipamentos [30, 61].

As plataformas tradicionais oferecem suporte eficiente para sistemas de alta performance, contudo, quando a demanda por volume, variedade, velocidade ou variabilidade aumenta ao extremo, estas soluções se tornam excessivamente caras. Os custos de adaptação dos sistemas tradicionais aumentam drasticamente à medida em que aumenta a escala [78]. Em função desta característica, os principais fornecedores de soluções de tecnologia estão oferecendo soluções de *big data* em conjunto com suas ferramentas tradicionais [47, 91, 107].

A velocidade de mudança é outra questão a se observar. Um *cluster big data* pode ser provisionado na nuvem em questão de minutos, respondendo a demandas como consultas *ad hoc* ou novas APIs de forma rápida. Neste cenário, as tecnologias de *big data* surgem como uma alternativa em função de sua boa relação entre custo e benefício, velocidade de implantação e velocidade de adaptação às demandas dos sistemas modernos [62].

Contudo, por ser relativamente nova, a área de *big data* ainda carece de mão de obra especializada, pois a implantação dos serviços é complexa e não trivial, uma vez que envolve a utilização de processamento massivamente paralelo por meio de tecnologias recentes. Há ainda os desafios técnicos relacionados com a tolerância a falhas, a escalabilidade, a qualidade e a heterogeneidade dos dados [78].

Para endereçar estes problemas, este trabalho propõe uma arquitetura de software que combina as facilidades da computação em nuvem e a performance dos sistemas de *big data*, como descrito na Seção 4.2.

4.2 Projeto da Arquitetura

Uma arquitetura de software é construída para atender a determinados requisitos. O projeto de uma arquitetura especializada em sistemas de *big data* é semelhante, pois tem de atender a requisitos pré-definidos. Entretanto, essas demandas são diferentes dos sistemas tradicionais.

Assim, esta seção apresenta o projeto da arquitetura para *big data*, baseando-se no roteiro proposto por Rozanski e Woods [113], com adequações para garantir o estado da arte na integração dos recursos de *big data* e de computação em nuvem. A arquitetura proposta usa e estende os modelos definidos por Bhagattjee [23], Chang [26, 28], Dukaric e Juric [44], Liu *et al.* [84], Zhang *et al.* [130].

4.2.1 Escopo

O **escopo** de uma arquitetura está relacionado com os objetivos e com as dificuldades enfrentadas pelos *stakeholders* para criar e usar efetivamente sistemas de *big data*. Como visto ao longo da dissertação, a criação desta classe de sistemas é complexa e faltam guias para a sua implantação em nuvem privada.

Assim, a arquitetura proposta oferece suporte para dois modos de processamento: *batch* e tempo real. O processamento *batch* será utilizado para análises sobre dados históricos, no qual a necessidade de acessar seu resultado não tem limitação rígida de tempo, ou seja, é possível esperar minutos ou horas pela entrega do resultado. O processamento em tempo real será usado para aplicações que demandam resposta imediata para a solicitação, como é o caso da integração entre sistemas.

4.2.2 Interesses

Os **interesses** da arquitetura incluem componentes para provisionamento, configuração, medição de uso, monitoramento e redimensionamento de recursos computacionais em sistemas de *big data*. A arquitetura dispõe de mecanismos para instanciar automaticamente *clusters big data* e bancos NoSQL por meio de serviços disponibilizados na nuvem. Como visto anteriormente, este é um modelo de serviço conhecido como *Big Data as a Service* (BDaaS).

Adicionalmente, a arquitetura usa o *datacenter* das organizações, funcionando como uma extensão para as suas capacidades. Consequentemente, representará uma alternativa às soluções disponíveis nos provedores de nuvem pública, que têm um custo associado.

Os componentes internos de cada camada podem ser implementados, evoluídos e gerenciados de forma independente. Assim, ferramentas com recursos específicos podem ser incluídas ou substituídas com pouco impacto no restante da estrutura.

4.2.3 Princípios

O projeto de sistemas *big data* é mais complexo que os projetos tradicionais, principalmente, porque envolve processamento distribuído. As dificuldades incluem não apenas as

tecnologias, mas também as técnicas de processamento e de armazenamento, que devem ser revistas neste novo contexto de aplicações *data-intensive*.

Para orientar a criação de sistemas de *big data*, Chen e Zhang [30] propuseram sete princípios:

1. Boa arquitetura e bons *frameworks* são prioritários: Existem muitas arquiteturas distribuídas para *big data* e cada uma usa estratégias diferentes para processamento em tempo real e em *batch*. Nestes casos, o NoSQL e o MapReduce são, respectivamente, as principais alternativas;

2. Suporte para vários métodos de análise: A ciência de dados se desenvolve rapidamente e envolve diversas técnicas que precisam estar suportadas pelas novas arquiteturas, como *data mining*, estatística, *machine learning* etc;

3. *No size fits all*: Ou seja, não existe uma solução que atenda a todas as situações, uma vez que cada tecnologia tem limitações. Deve-se escolher a ferramenta adequada para cada técnica e situação;

4. A análise deve estar próxima de onde estão os dados: Este é um princípio que atende a requisitos de máxima performance. Entretanto, a alta velocidade de transferência atualmente permite uma nova configuração, no qual os dados não precisam estar adjacentes ao processamento. Esta nova possibilidade permite o desacoplamento entre dados e processamento por meio de *data lakes*, como será detalhado na Seção 4.4.3;

5. O processamento precisa ser distribuído para análise em memória: O processamento massivamente paralelo, do inglês *Massively Parallel-Processing* (MPP), é uma das bases dos sistemas de *big data*, no qual os dados são acumulados no sistema de armazenamento do *datacenter*, mas devem ser particionados para permitir o processamento paralelo;

6. O armazenamento precisa ser distribuído para retenção em memória: Este princípio está diretamente ligado ao anterior e é atendido pelas mesmas tecnologias, ou seja, as ferramentas para MPP costumam dividir os dados em blocos de tamanho variável na memória;

7. É necessário um mecanismo para coordenar as unidades de dados e de processamento: Para garantir tanto a escalabilidade quanto a tolerância a falhas é necessário dispor de ferramentas para coordenação dos processos entre os recursos computacionais da nuvem.

4.2.4 Restrições

Como **restrição** as ferramentas para implementação devem ser unicamente *free and open-source* (FOSS), para garantirem que a solução possa ser usada em órgãos do governo ou empresas privadas, sem as limitações do custo de aquisição. Durante a implantação dos sistemas é possível escolher entre as principais implementações de soluções de *big data*, como Cloudera [32], HortonWorks [64], Apache Spark [12], além da versão original do Apache Hadoop [6].

Não estão incluídas na arquitetura questões referentes à segurança, acessibilidade, internacionalização, regulação e usabilidade [113]. A segurança dos dados estará a cargo dos administradores da infraestrutura. Por se tratar de um sistema sem interface gráfica para o usuário comum, não serão disponibilizados recursos para acessibilidade. O projeto da arquitetura prevê sua utilização em organizações públicas e privadas, mas apenas em língua portuguesa. Além disso, a legislação sobre regulação não se aplica porque os dados usados nos experimentos não foram publicados além dos limites legais. A última restrição diz respeito à visualização de dados. A arquitetura não dispõe de mecanismos para visualização ou geração de relatórios e gráficos.

4.2.5 *Stakeholders* do Sistema

Os *stakeholders* são as partes interessadas na arquitetura. Não está prevista uma estratégia para o engajamento dos *stakeholders*, conforme sugerido por [113]. Neste caso, considera-se que as organizações estejam comprometidas com a implantação do sistema de *big data*. Para a arquitetura de sistemas de *big data*, os interessados são:

- Cientistas de Dados: quem executa consultas *ad hoc* no mecanismo de consulta ou análises de dados de predição, de estatística e de *machine learning*;
- Desenvolvedores de Software: quem cria os sistemas;
- Sistemas Corporativos: sistemas e bancos de dados em operação na organização ou fora dela;
- Administradores da Infraestrutura: responsáveis por manter o ambiente do *data-center*, incluindo servidores, armazenamento, rede e banco de dados;

4.2.6 Cenários de Uso

Os cenários de uso da arquitetura proposta são descrições das funcionalidades que o sistema deve apresentar. Assim, os cenários mostram de que forma o sistema pode ser usado. Na arquitetura proposta, alguns cenários são independentes, como é o caso de Criar

cluster de *big data*, Criar banco NoSQL, Liberar recurso computacional e Redimensionar *cluster* de *big data*. Nestes casos, o usuário pode simplesmente executar a ação. Contudo, algumas situações dependem de pré-condições, como o caso de uso Executar análise de dados, que precisa de um *cluster* de *big data* provisionado para sua realização.

O diagrama de caso de uso, que mostra graficamente os cenários envolvidos na arquitetura, pode ser conferido na Figura 4.1. Assim, estão listadas e delimitadas as situações típicas de um sistema de *big data* em ambiente corporativo que serão atendidas pela arquitetura proposta.

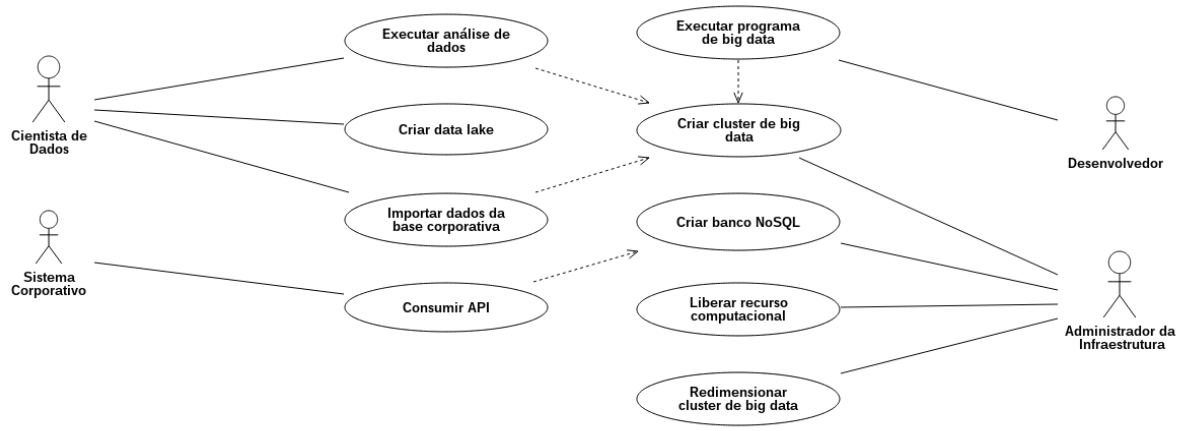

Figura 4.1: Diagrama de Caso de Uso.

4.2.7 Visões Arquiteturais

As visões arquiteturais são usadas para mostrar diferentes aspectos do sistema de *big data* de forma abstrata. A formalização da documentação segue o modelo proposto em [113], assim, foram selecionadas as visões de contexto, funcional, implantação e operacional. A combinação destas visões ilustra com clareza como deve ser construído e implantado um sistema de *big data* na arquitetura de BDaaS proposta. Os detalhes técnicos estão detalhados na Seção 4.3.

Visão de Contexto

A Visão de Contexto apresenta a arquitetura de uma perspectiva conceitual, na qual é ilustrado o ambiente operacional do projeto e mostra o que foi incluído, e o que não foi nos limites da arquitetura de BDaaS. Assim, como mostrado na Figura 4.2, de um lado estão as fontes de dados, no centro a nuvem privada e o BDaaS, e à direita os usuários do sistema.

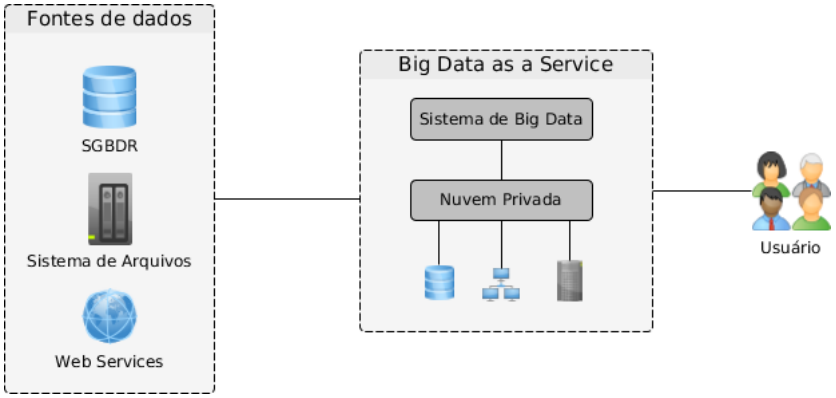

Figura 4.2: Visão de Contexto da Arquitetura.

As fontes de dados são compostas por SGBDRs, sistemas de arquivos ou *web services* pré-existentes na organização. Após a importação destes registros no sistema de *big data*, os dados estão disponíveis para processamento ou armazenamento e, em seguida, estão acessíveis aos usuários.

A arquitetura de BDaaS fornece uma plataforma para os sistemas de *big data* na forma de serviço, ou seja, o sistema pode ser implantado diretamente sem preocupação com armazenamento, processamento e rede, pois esta é a responsabilidade do BDaaS.

Visão Funcional

A Visão Funcional descreve os usos, os componentes, as interfaces, as entidades externas e as principais interações entre eles [113]. Assim, seguindo essa definição, a Figura 4.3 apresenta a Visão Funcional da arquitetura proposta por meio de um diagrama de componentes.

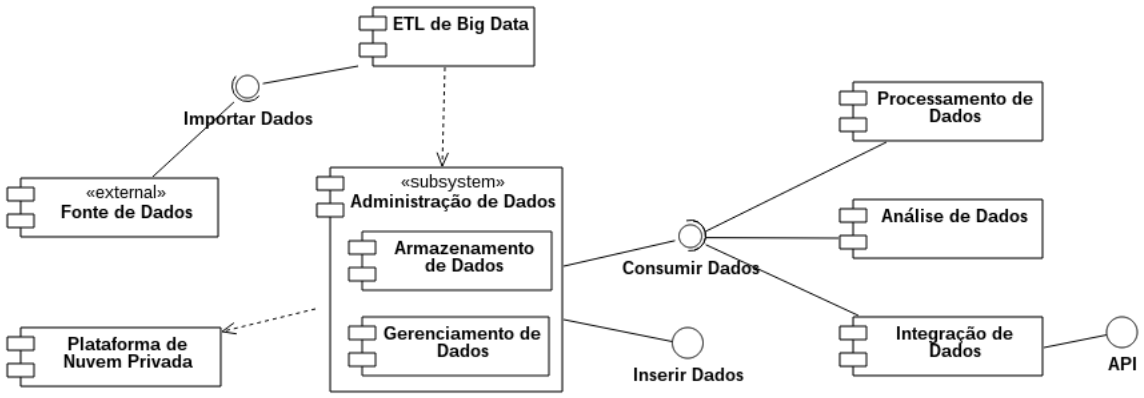

Figura 4.3: Diagrama de Componentes da Arquitetura.

O componente de **Fonte de Dados** é uma entidade externa que representa, como o nome sugere, qualquer mecanismo que forneça dados corporativos, e inclui os SGBDRs, os sistemas de arquivos e os *web services*.

O **ETL de *Big Data*** realiza o processamento necessário para converter os dados de seu formato de origem para os formatos suportados pelo mecanismo de armazenamento de *big data*. Este componente está detalhado na Seção 4.4.

O sub-sistema de **Administração de Dados** é um componente lógico que envolve dois componentes. O primeiro deles é o serviço de **Armazenamento de Dados**, responsável por persistir os dados no sistema de arquivos distribuído. O componente de **Gerenciamento de Dados** é constituído pelo banco de dados NoSQL. Este sub-sistema dispõe de uma interface para que novos registros sejam inseridos sem a necessidade de passar pelo ETL de *Big Data*. A outra inferface permite que os demais componentes consumam diretamente os dados.

O **Processamento de Dados** é composto pelos mecanismos para processamento *batch* e tempo real, além de oferecer suporte para a implantação de programas de *big data*, como Hadoop e Spark, a partir de seus respectivos *frameworks*.

O serviço de **Integração de Dados** é uma funcionalidade presente nas arquiteturas de *big data* mais recentes. Ele oferece uma API para que entidades externas consumam os dados do sistema. A API é construída a partir de um banco NoSQL e disponibilizada por meio de *web services*, uma tendência na área de *big data* e computação em nuvem que está detalhada na Seção 4.4.

O último serviço é o de **Análise de Dados**, que combina mecanismos de consulta *ad hoc*, de estatística e de algoritmos de *machine learning*. Estas funcionalidades são usadas pelos cientistas de dados e fazem parte de uma área conhecida como *big data analytics*.

Visão de Implantação

A Visão de Implantação mostra o ambiente onde o sistema será instalado, e mostra o hardware e o software necessário para sua execução, ou seja, nesta visão são definidos os tipos de equipamentos necessários, os softwares e os requisitos de rede básicos para implantar o sistema [113].

Esta visão mostra dois diagramas, um para a implantação da plataforma de nuvem privada e outro para a implantação do sistema de *big data*. A Figura 4.4 mostra os equipamentos necessários para a implantação de nuvem privada. A única forma de acesso aos recursos da nuvem é feito por meio da interface de acesso.

Os sistemas de *big data* criados a partir desta arquitetura são implantados por meio da plataforma de nuvem privada, como pode ser visto na Figura 4.5. Os servidores representados no diagrama podem ser VMs ou servidores físicos (*bare metal*). Para a

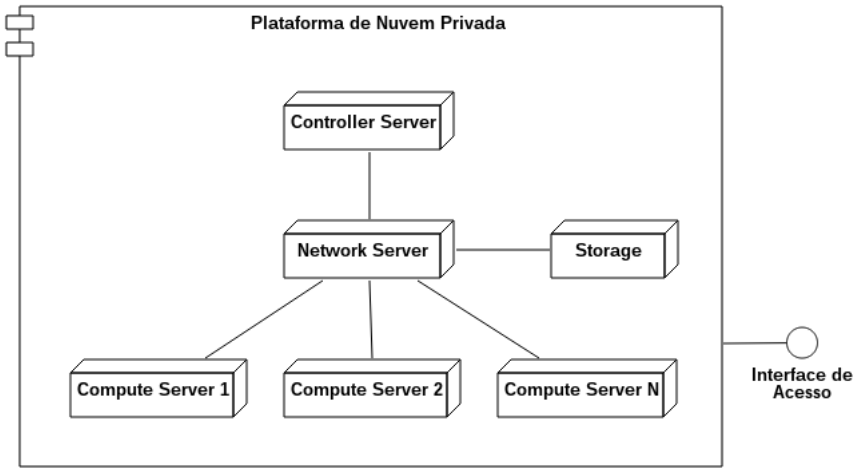

Figura 4.4: Diagrama de Implantação da plataforma de nuvem privada.

criação de provas de conceito é possível utilizar um único servidor para a plataforma de nuvem privada. Neste caso, a performance não pode ser avaliada, porque os recursos computacionais são limitados. Apesar disso, é possível validar se as funcionalidades foram atendidas.

Existem quatro funcionalidades principais previstas na arquitetura, que atendem às demandas dos sistemas de *big data*. As funcionalidades são (i) *Cluster* de *Big Data*, no qual residem os serviços de Armazenamento e Análise de Dados. O (ii) Servidor de API engloba o serviço de Integração de Dados, enquanto o (iii) Servidor NoSQL engloba o Gerenciamento de Dados. Por fim, o (iv) *Storage* oferta novamente o serviço de **Armazenamento de Dados**, com a diferença de que nesta configuração não existe a necessidade do *cluster* de *big data*. Esta técnica é vista em detalhe na Seção 4.4.

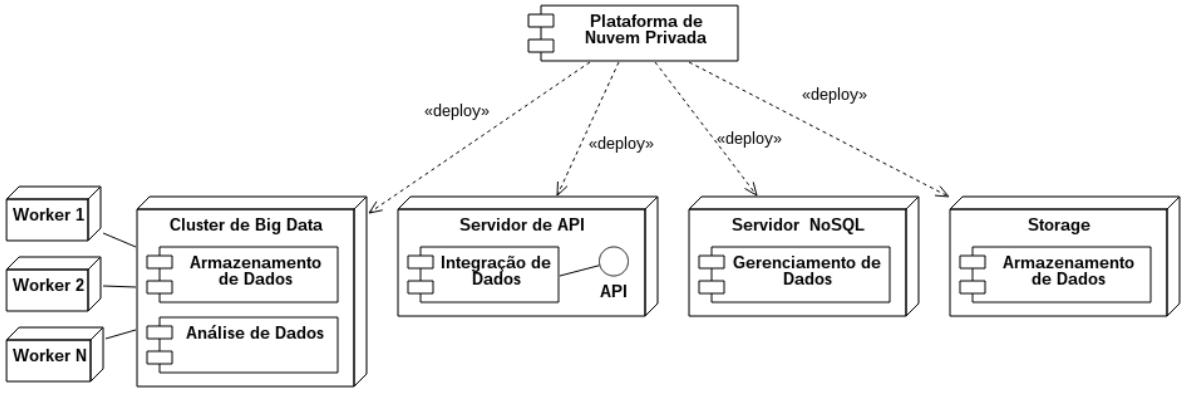

Figura 4.5: Diagrama de Implantação do Sistema de *big data*.

Como visto nesta seção, a arquitetura é composta por elementos de software e de hardware. A Visão de Implantação sugere o uso de tabelas para relacionar esses elementos com o software que precisa ser instalado. Assim, a Tabela 4.1 faz o mapeamento entre os componentes da arquitetura e os principais softwares disponíveis para implementação.

Tabela 4.1: Dependências de Software dos Componentes.

Componente	**Requisito**
Plataforma de Nuvem Privada	OpenStack Pike CentOS 7 KVM
Big Data Access Tool	Java 8[1] Apache Sqoop[2] Apache Kafka[3]
Armazenamento de Dados	Apache HDFS OpenStack Swift
Gerenciamento de Dados	Apache Cassandra Apache HBase
Processamento de Dados	Apache Hadoop Apache Spark Cloudera Hortonworks
Integração de Dados	Spring Boot 2.0
Análise de Dados	Apache Hive Apache Spark Hue

Visão Operacional

A Visão Operacional descreve a operação, a administração e o suporte do sistema em seu ambiente de produção. Aqui são definidas estratégias para instalação, atualização, migração, monitoramento, *backup/restore* e configuração do sistema [113].

A operação do sistema pode ser verificada na Figura 4.6, que mostra as ligações entre os principais elementos da estrutura. O ponto de partida são os usuários que, por meio da Interface de Acesso do sistema, podem solicitar os serviços disponíveis. Depois desta solicitação, a Plataforma de Nuvem verifica quais recursos são necessários para executar o serviço. O último passo é o provisionamento das VMs e a instalação dos softwares por meio da Virtualização.

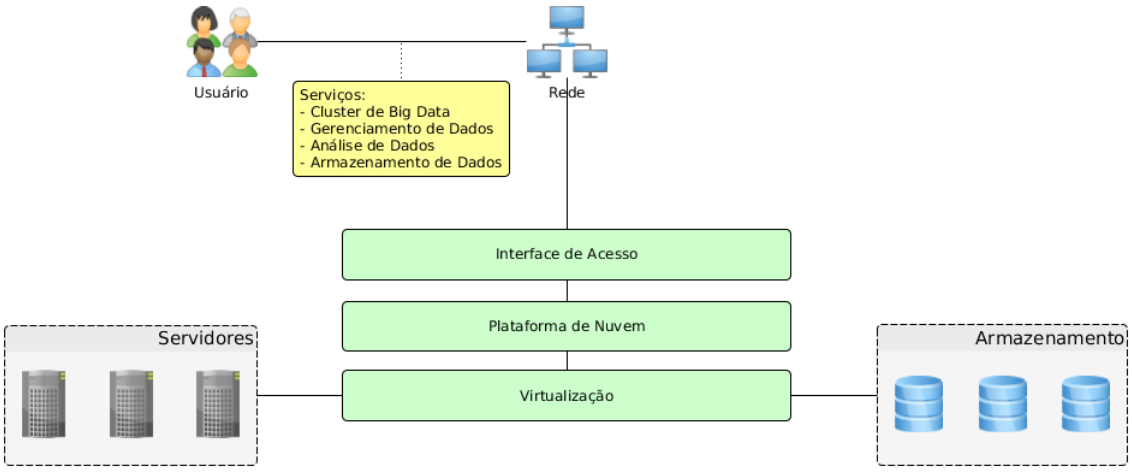

Figura 4.6: Visão Operacional da Arquitetura.

4.3 Arquitetura em Camadas

A arquitetura de BDaaS proposta obedece ao estilo arquitetural em camadas, ou seja, a arquitetura é dividida em módulos lógicos e independentes. As camadas representam uma abstração para os *stakeholders* do sistema de *big data*, escondendo as complexidades de implantação. Com essa estratégia, os *stakeholders* podem focar apenas na realização de suas atividades, sem entrarem em detalhes de implantação das ferramentas.

Cada camada tem uma responsabilidade bem definida, mesmo que sua implementação compartilhe o uso de ferramentas ou mesmo de hardware. Por exemplo, duas camadas podem usar o mesmo servidor ou o mesmo *framework*, mas suas responsabilidades continuam independentes.

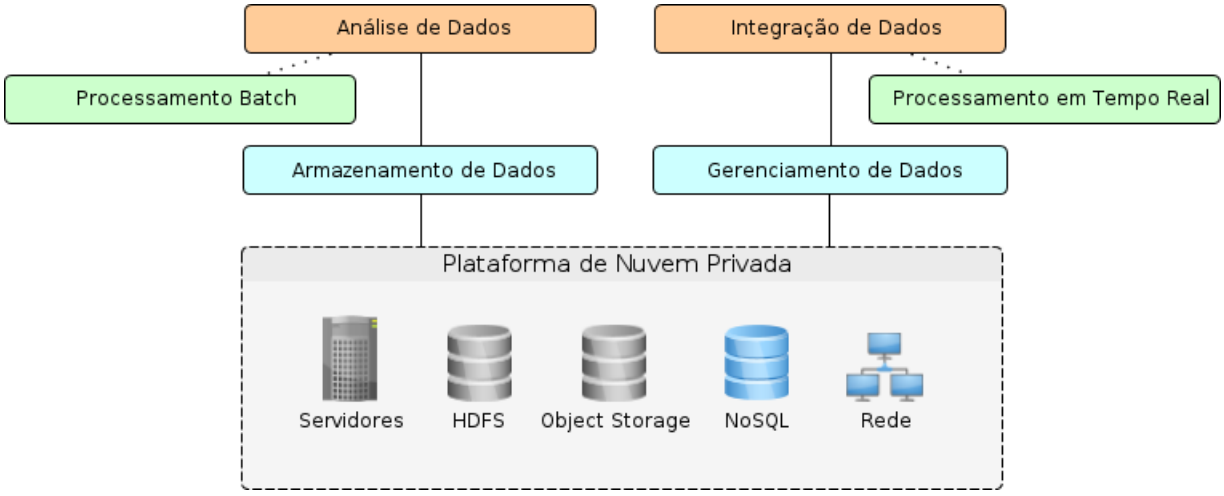

Figura 4.7: Arquitetura proposta para *Big Data as a Service* em Nuvem Privada.

A Figura 4.7 mostra a visão conceitual e os principais relacionamentos entre as camadas. A camada de Análise de Dados, com o uso da camada de Processamento *Batch*, executa as consultas ou os programas com os registros disponíveis na camada de Armazenamento de Dados, que acessa os recursos virtualizados da Plataforma de Nuvem Privada. Da mesma forma, a camada de Integração, por meio do Processamento em Tempo Real, acessa os serviços da camada de Gerenciamento de Dados, que usa, igualmente, os recursos virtualizados da nuvem.

4.3.1 Camada de Integração de Dados

A Camada de Integração de Dados está na parte mais externa da arquitetura, na qual estão definidas as listas de dados com acesso controlado por meio da API de *web services*, fornecendo um mecanismo de comunicação com baixo acoplamento e alta performance. Os usuários com acesso a esta camada são os sistemas de informação internos da própria organização ou de órgãos externos.

O uso de uma API Web confere flexibilidade à arquitetura, que se torna agnóstica em relação à linguagem de programação ou tecnologias adotadas no lado do usuário. Segundo Pautasso [110], as arquiteturas orientadas a serviço promovem o projeto de sistemas distribuídos e integrados através da composição de serviços autônomos e reusáveis.

A API Web permite as operações de inclusão, alteração, exclusão e consultas básicas. Esta camada atende primariamente ao processamento em tempo real e pode funcionar como um mecanismo para injetar os dados para o processamento *batch*, como visto na Figura 4.8.

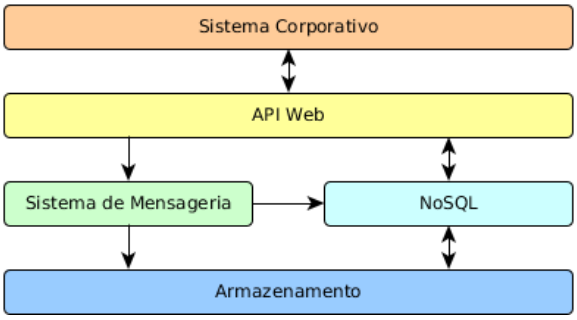

Figura 4.8: Camada de Integração de Dados.

O fluxo de utilização da Integração de Dados começa com o Sistema Corporativo fazendo um acesso à API Web. A partir da API Web existem dois caminhos: (i) ingestão de dados no Sistema de Mensageria, de forma assíncrona; e (ii) consulta ao NoSQL. Deve-se notar que as operações com o Sistema de Mensageria são unidirecionais, uma vez que o objetivo da proposta é permitir a inserção de novos registros por este caminho como alternativa para melhoria da performance [26].

Para criar a API Web foi usado o Spring Boot [118], um software amplamente adotado para a criação de *microservices*. *Microservices* são uma tendência recente e formam a base para a criação de APIs Web. A estratégia é dividir e conquistar, ou seja, é feita a decomposição do sistema em pequenas funcionalidades encapsuladas em sub-sistemas isolados e independentes que se comunicam por meio de um protocolo, como o HTTP [69].

No caso da mensageria, o sistema selecionado foi o Apache Kafka [16], uma escolha que atende à arquitetura proposta porque ele suporta processamento distribuído e é escalável, além de ser facilmente integrado com *clusters* Hadoop. Assim, para a inserção de registros em alta velocidade é usada a combinação da API Web com o sistema de mensageria, ou seja, o sistema externo envia os dados para um *microservice* no Spring Boot, que por sua vez redireciona os dados para o Kafka. O Kafka processa os dados de entrada e grava o registro no formato de destino, que é o banco NoSQL, o HDFS ou o *object storage*.

4.3.2 Camada de Análise de Dados

A Camada de Análise de Dados permite ao usuário do BDaaS executar consultas nas bases de dados a partir das linguagens proprietárias dos *frameworks*, ou por meio de comandos SQL para aproveitar o conhecimento dos profissionais de tecnologia. A possibilidade de executar consultas SQL é uma funcionalidade dos novos *frameworks* de *big data*,

A principal característica desta camada é a abstração de toda a complexidade de implantação de infraestrutura, carregamento e tratamento dos dados da organização. Assim, por meio de uma interface *web*, o usuário executa consultas diretamente no sistema de armazenamento distribuído, e o processamento é realizado em modo *batch*, por se tratar de análises históricas sobre todos os dados gravados.

Assim, são usadas duas ferramentas. A primeira é o Apache Hive [10], que é uma ferramenta de *data warehouse* que foi selecionada porque tem suporte a SQL e também porque facilita a leitura, a escrita e o gerenciamento de grandes conjuntos de dados armazenados no HDFS. O Hive tem conectividade através de Java Database Connectivity (JDBC)[4], o que permite a integração com diversas fontes de dados.

O Hive será usado para consultas *ad hoc*, de forma que os profissionais possam executar comandos SQL diretamente no Hadoop, sem a necessidade de criação de programas MapReduce, uma conveniência que agiliza a resposta para o cliente. A segunda ferramenta para análise de dados é o Apache Spark, um *framework* para processamento *in-memory* que foi selecionado porque combina capacidades de processamento paralelo, consultas SQL, tempo real, *machine learning* e grafos.

[4]http://www.oracle.com/technetwork/java/javase/jdbc/index.html

4.3.3 Camada de Gerenciamento de Dados

A Camada de Gerenciamento de Dados é formada por um banco de dados NoSQL com características de escalabilidade, e suporte a tabelas com grandes volumes de dados. Assim, ela será usada para situações de gravação e de leitura intensiva de registros individuais e aleatórios. Nesta camada, o usuário tem acesso direto aos dados e pode realizar consultas ou alterações nos registros. As operações de carregamento de dados são realizadas por meio de uma ferramenta de ETL de *big data*, a ser discutida na Seção 4.4.

O banco de dados escolhido foi o Apache Cassandra [5] em função da performance, documentação de boa qualidade e estabilidade. Uma característica importante do Cassandra é que ele tem recuperação rápida de registros a partir da chave-primária. Outro ponto considerado é que o Cassandra tem *drivers* para Hadoop e pode ser usado como uma de suas fontes de dados.

4.3.4 Camada de Processamento *Batch*

A Camada de Processamento *Batch* executará programas Hadoop escritos pelo usuário. Neste caso, o usuário precisa ser especialista em programação distribuída e Hadoop. A aplicação deverá ser submetida para execução em *clusters* disponíveis na plataforma, ou um novo *cluster* deve ser provisionado para esta execução.

A implementação desta camada será feita com o Apache Hadoop e o seu ecossistema, principalmente o *MapReduce*, por ser um *framework* adequado a este contexto de processamento *batch* [76]. Contudo, deve-se observar que a latência esperada para a conclusão das análises nesta camada é alta. O tipo de aplicação mais adequado para esta camada é aquela que precisa processar dados históricos de uma só vez.

4.3.5 Camada de Processamento em Tempo Real

A Camada de Processamento em Tempo Real oferece a opção de processamento contínuo sobre dados inseridos constantemente no sistema e que precisam de uma resposta rápida. Esta camada é um complemento à Camada de Processamento *Batch* e o usuário precisa ter conhecimento sobre programação para usá-la.

A implementação é feita por *frameworks* de processamento *in-memory* e de mensageria, nos quais é possível publicar e consumir fluxos de dados. O processamento é realizado à medida em que os dados são inseridos e são disponibilizados recursos de computação distribuída como tolerância a falhas, escalabilidade e alta performance. Desta forma, para a inserção de novos registros serão criadas filas de mensagens usando o Kafka [16] para processamento em tempo real e inclusão dos registros no NoSQL e no sistema de armazenamento de dados (HDFS e Swift).

4.3.6 Camada de Armazenamento

A Camada de Armazenamento é responsável pela guarda final dos dados de forma permanente ou temporária, de acordo com a demanda do usuário. Para isso, são disponibilizados dois tipos diferentes de armazenamento: (i) *object storage* com o OpenStack Swift [105] e (ii) HDFS, que nativamente faz parte do *cluster* Hadoop. Entretanto, o Hadoop dispõe de integração com outros sistemas de arquivos, como o S3 [4] ou o Swift, sendo que este foi a ferramenta selecionada na arquitetura proposta porque tem integração com o OpenStack.

O *object storage* é o principal mecanismo de armazenamento para as análises de dados, especialmente para a criação de *data lakes*, tema que será detalhado na Seção 4.4.3. O HDFS é outra opção para o armazenamento, entretanto, é uma solução acoplada ao próprio *cluster* Hadoop, o que vai contra a proposta da arquitetura de ter serviços com baixo grau de acoplamento. O último tipo de armazenamento é em blocos, que é usado para armazenar as VMs.

4.3.7 Camada da Plataforma de Nuvem Privada

A Camada da Plataforma de Nuvem Privada é responsável pelo gerenciamento dos recursos computacionais. O usuário poderá iniciar ou parar a execução dos serviços por meio da interface de acesso, sem conexão direta com os recursos de hardware ou tecnologias de implantação.

A implantação da camada será feita por meio do OpenStack [104, 115, 123], um software gerenciador de recursos computacionais para criação de nuvens privadas ou híbridas. Esta camada é responsável pelo provisionamento dos recursos físicos, incluindo processadores, memória, rede e armazenamento. O provisionamento do *cluster* de *big data* e de NoSQL está fora do escopo padrão do OpenStack e será implementado com módulos específicos, chamados de OpenStack Sahara [39, 101] e OpenStack Trove [106].

A alocação de recursos seguirá as políticas definidas pela organização. Essas políticas incluem quotas para o número de processadores, de armazenamento e de largura de banda da rede, uma vez que o número de equipamentos é limitado. As políticas de utilização devem levar em consideração a atualização, a falha dos recursos físicos e a rotina de recuperação nestes casos.

Os recursos computacionais são virtualizados e compartilhados por todos os serviços. A virtualização é realizada pelo KVM na arquitetura proposta. A plataforma de nuvem suporta o provisionamento de recursos *bare metal*, ou seja, é possível provisionar um *cluster* com servidores físicos. Contudo, a forma mais comum de provisionamento é com o uso de máquinas virtuais.

Por fim, o monitoramento é responsável por verificar as condições de funcionamento e a utilização do sistema. É um módulo opcional, contudo, importante para a coleta de métricas que podem ser usadas para otimização dos recursos.

Os Administradores da Infraestrutura são os usuários desta funcionalidade, e têm acesso a métricas sobre a disponibilidade e a utilização dos serviços. O objetivo está relacionado com a qualidade do serviço ofertado, pois permite o monitoramento de falhas, das indisponibilidades, da subutilização e da sobrecarga dos recursos. Desta forma, o monitoramento da nuvem pode contribuir para a elasticidade e a escalabilidade da arquitetura.

4.4 Técnicas para Construção de Sistemas de *Big Data*

A descrição de uma arquitetura deve detalhar as melhores práticas para a construção dos sistemas [113], o que é especialmente importante em uma área tão recente e complexa quanto o *big data* [89]. A construção de sistemas de *big data* baseados em nuvem computacional tem técnicas específicas para dimensionamento, carregamento, armazenamento, modelagem e integração de dados.

A implantação de infraestrutura para sistemas de *big data* demanda grande esforço das equipes de tecnologia. As dificuldades incluem: (i) instalação e configuração do *cluster* de *big data* e de bancos NoSQL; (ii) dimensionar os recursos e atender a mudanças na demanda de processamento; e (iii) disponibilização dos serviços de dados. Assim, são listadas técnicas, ambientes e tecnologias usadas em pontos-chave do *datacenter* e como elas podem ser usados para construir a arquitetura de soluções de análise de *big data*. Além dos aspectos técnicos, são abordadas recomendações e tendências não técnicas motivadas pelas pesquisas mais recentes na área.

Apesar de ser um tópico amplamente pesquisado, a análise de *big data* só é possível por meio de uma infraestrutura complexa. Os desafios para sua implementação com sucesso na organização costumam incluir altos custos com a aquisição de licenças de software, consultoria e hardware. Parte dessa complexidade é absorvida pela computação em nuvem, que oferece a IaaS como uma facilidade para provisionamento de recursos no *datacenter* [21].

4.4.1 Dimensionamento de Recursos

O dimensionamento de recursos para sistemas de *big data* em nuvem é tema de diversas pesquisas [117, 120, 130], incluindo o uso de algoritmos preditivos [124] e o provisionamento automático de *clusters* Hadoop [39]. Em muitas empresas é comum encontrar

clusters com dezenas de servidores, entretanto, este tipo de instalação tende a ser superdimensionada para atender aos picos de processamento.

Desta forma, deve-se evitar a criação de grandes *clusters*, sempre que possível, uma vez que o provisionamento de muitas instâncias tende a gerar ociosidade de recursos. Por isso, para a criação de um *cluster* ou banco NoSQL, é necessário verificar o tempo total do processamento. Neste ponto, existem duas situação distintas:

- Processamento constante: as VMs, o *cluster* ou o NoSQL ficam provisionadas por tempo indeterminado;
- Processamento efêmero: provisionamento de *cluster* pequeno por tempo determinado com acesso ao *data lake*.

Observa-se que a implantação deste tipo de infraestrutura demanda grande esforço das equipes de tecnologia. As dificuldades incluem [96]:

- Esforço manual para a implantação, o gerenciamento, a atualização e o *backup* dos *clusters*, o que gera atraso nas análises desenvolvidas pelos cientistas de dados;
- Dificuldade para gerenciar a performance, uma vez que as aplicações entram em produção e precisam ser dinamicamente escalonadas para atender à carga real dos usuários, que pode ser maior ou menor que o previsto;
- Baixa utilização dos recursos computacionais, com ociosidade de hardware e de software;
- Criação de repositórios de dados não compartilhados que necessitam de intervenção manual para transferência das informações.

Com a arquitetura proposta, a recomendação para atender a demanda de processamento em larga escala é usar diversos *clusters* menores de *big data*, um para cada tipo de carga de trabalho, uma vez que a maioria dos programas Hadoop são executados em *datasets* com menos de 100 GB [19]. Esta situação foi verificada nos experimentos realizados, uma vez que o tamanho do *dataset* completo importado do SGBDR ocupa aproximadamente 80 GB no HDFS em formato compactado.

Como ponto de partida, nestes casos, pode ser usado um *cluster* com até três nós, cada um com 16 GB de memória, totalizando 48 GB. Após o fim das análises de dados, os recursos do *cluster* podem ser liberados. Para um banco NoSQL o ponto de partida é o provisionamento de uma VM separada com 16 GB de memória, na qual é instalado apenas o Cassandra. Neste caso os recursos não são liberados, uma vez que a duração da utilização do banco NoSQL é indeterminada.

4.4.2 Armazenamento na Nuvem

Nas versões iniciais do Hadoop, as análises de dados eram executadas utilizando dados do sistema de arquivos local do *cluster*, porque o HDFS é otimizado para esta finalidade. Entretanto, para sistemas baseados em nuvem, esta abordagem não é a mais eficiente, porque o HDFS não foi projetado para este cenário. Como os dados estão diretamente ligados ao *cluster*, há limitação ou mesmo impossibilidade do uso das características da nuvem.

Assim, por exemplo, considerando um *cluster* Hadoop, caso o usuário necessite de mais armazenamento, não é possível, facilmente, aumentar a capacidade de armazenamento, porque apenas a equipe de operações do *datacenter* tem essa capacidade. Da mesma forma, quando o *cluster* é liberado, seus dados geralmente são apagados. Para fazer novas análises sobre esse *dataset* que foi excluído, os dados precisam ser copiados novamente para um *cluster*.

Uma possível solução para este problema é o uso da tecnologia de *object storage*, que separa o processamento do armazenamento dos dados. O *object storage* (ObS), ou *object storage device* (OSD) armazena os dados como objetos de tamanho variável, ao contrário do armazenamento tradicional de arquivos em blocos [49]. Assim, são características do *object storage*: a durabilidade, a alta disponibilidade, a replicação, e a elasticidade, permitindo que a capacidade de armazenamento seja virtualmente infinita. No *object storage* cada item armazenado é um objeto definido por um identificador único, oferecendo uma alternativa ao modelo de arquivos baseados em blocos de dados.

Por causa dessas facilidades, o armazenamento de dados na nuvem pode ser feito por meio do *object storage*. Seguindo essa tendência, os principais provedores de nuvem têm suas implementações de *object storage*, como o AWS S3 [4], o Oracle Object Storage [108], o Azure Blob Storage [92] e o Google Cloud Storage [59]. No Openstack, o módulo de *object storage* é o Swift, no qual podem ser executadas as análises de dados [114], seguindo a arquitetura proposta nesta dissertação.

Assim, considerando a situação na qual a demanda de recursos oscila, a abordagem mais indicada para persistência de *big data* na nuvem é a utilização de armazenamento baseado em objetos (*object storage*), que é o recomendado para análise *big data* e *data mining* [123].

4.4.3 *Data Lake*

Data lakes são repositórios centralizados de dados corporativos, incluindo dados estruturados, semi-estruturados e não estruturados. Esses dados estão, geralmente, em seu formato nativo e armazenados em sistemas de arquivos de baixo custo e alto desempenho,

como o HDFS ou o *object storage* [43] [94]. A proposta do *data lake* é diferente de um *data warehouse* (DW). No DW os dados estão processados e estruturados para a consulta, e a estrutura é definida antes da ingestão no sistema, por meio de rotinas de ETL. Esta técnica é chamada de *schema-on-write*, uma tarefa que não é tecnicamente difícil, mas consome tempo.

No *data lake* os dados estão em seu formato original, com pouca ou nenhuma transformação, e a estrutura dos dados é definida durante sua leitura, uma técnica conhecida como *schema-on-read*. Os usuários podem definir e redefinir rapidamente os esquemas dos dados durante o processo de leitura dos registros. Com isso, o ETL é executado a partir do próprio *data lake* [50].

O provisionamento e a configuração do *data lake* são executados pela plataforma de nuvem privada, com o módulo OpenStack Swift. O Swift tem integração com o Hadoop e o Spark, de forma a permitir análises de dados com os principais formatos de arquivos: SequenceFiles, Avro e Parquet [15, 17, 85].

A vantagem do *data lake* é a sua flexibilidade, que é ao mesmo tempo um problema, porque torna as análises complexas. Dessa forma, os usuários do *data lake* devem ser altamentes especializados, como os cientistas de dados e, excepcionalmente, os desenvolvedores. Existem ainda outros riscos na adoção de *data lakes*, como a garantia da qualidade, a segurança, a privacidade e a governança de dados, que são questões ainda abertas.

A utilização de *data lakes* demanda treinamento da equipe de análise de dados em novas ferramentas e técnicas, o que traz uma complexidade adicional. Contudo, sua adoção tem o potencial de diminuir os gastos com infraestrutura de armazenamento e software de banco de dados, uma vez que é possível usar hardware de baixo custo e software livre.

4.4.4 *Framework* de ETL de *Big Data*

O processo de *extract, transform and load* (ETL) é um dos primeiros passos para a operação do sistema de *big data*. De forma geral, o ETL compreende as tarefas de (i) validação dos dados de entrada, (ii) limpeza dos dados com a remoção de registros incorretos, (iii) padronização dos registros para adequação ao esquema de dados e (iv) indexação para suporte a buscas rápidas. A visão conceitual do *framework* de ETL pode ser conferida na Figura 4.9.

Os dados estão originalmente armazenados nas fontes de dados, como SGBDR, sistemas de arquivos ou *web services*. A primeira tarefa do ETL é extrair os registros e armazená-los em uma área temporária, chamada de *data staging*, onde são executadas as operações de limpeza dos registros. Em seguida, os dados são inseridos no dispositivo de

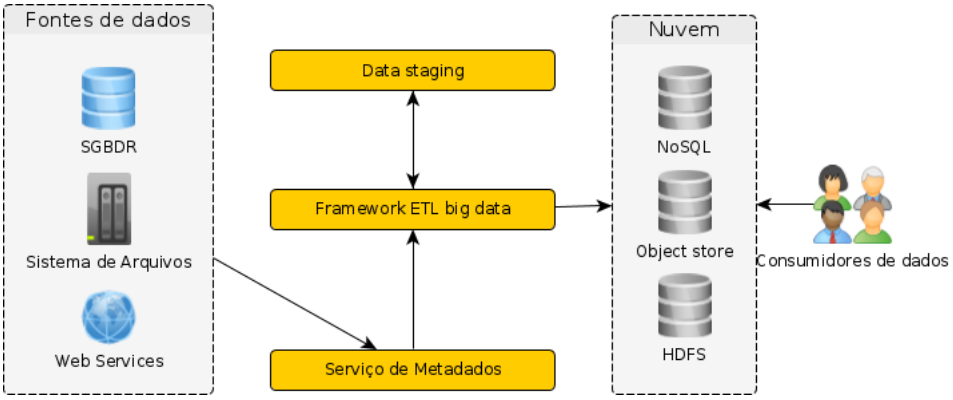

Figura 4.9: Framework para ETL de *big data*.

armazenamento na nuvem, que podem ser um banco de dados NoSQL, o *object store* ou o próprio HDFS de um *cluster* existente. A partir deste ponto os dados estão disponíveis para consumo dos usuários.

4.4.5 Modelagem NoSQL

O banco de dados NoSQL tem acesso rápido para leitura e escrita, suporta grandes volumes de dados e replicação, desta forma, é uma tecnologia adequada para sistemas de *big data*. Entretanto, há diferenças entre a modelagem de dados em bancos NoSQL e em bancos relacionais. Enquanto o modelo de um SGBDR é normalizado para evidar redundância de dados, os bancos NoSQL não usam normalização e os dados estão muitas vezes duplicados em diversas colunas e tabelas para garantir máxima performance [29].

Seguindo o princípio da desnormalização, as tabelas NoSQL em um banco colunar são criadas para responder a consultas em campos específicos. Para cada conjunto de campos pesquisáveis, é gerada uma nova tabela [29]. Por exemplo, considerando uma pesquisa definida por *p(ConjuntoDados, CampoA, CampoB)*, na qual ConjuntoDados se refere a uma fonte de dados e CampoA/CampoB são campos existentes, deve-se gerar uma tabela t1, ou seja, *t1 = p(ConjuntoDados, CampoA, CampoB)*.

De forma análoga, para uma consulta sobre outro campo, como o CampoC, deve-se gerar outra tabela, definida por *t2 = p(ConjuntoDados, CampoC)*, observando que ConjuntoDados é o mesmo, o que irá gerar duplicação de dados. Seguindo essa técnica, o *framework* de ETL da arquitetura proposta cria ao menos uma tabela NoSQL para cada tabela relacional, observando que os campos pesquisáveis são aqueles que fazem parte do índice da tabela.

4.4.6 Gerenciamento de API

O uso de APIs na Internet está se tornando o padrão para aplicações web, *mobile*, nuvem e *big data* [119]. As APIs facilitam a troca de dados e são usadas para integrar empresas, disponibilizar algoritmos, conectar pessoas e compartilhar informações entre dispositivos. Este novo modelo de negócio, chamado de economia das APIs, permite que as empresas se transformem em verdadeiras plataformas de dados, o que simplifica a criação de novos serviços, produtos e modelos de negócio [56].

As APIs são compostas por serviços independentes na forma de componentes reusáveis, que podem ser combinados para a criação da plataforma de dados. Por exemplo, uma empresa pode criar um novo serviço usando APIs de terceiros, como mapas, *machine learning*, geolocalização e pagamentos. Esses serviços são usualmente baseadas em REST e JSON/XML [52, 70, 110], assim, permitem o compartilhamento tanto de dados como de funcionalidades com ótima performance. Essa é a estratégia adotada por grandes provedores e usuários de API, como Netflix [99], Google [58], AWS [22] e eBay [45].

Neste contexto, é extremamente importante que uma arquitetura de *big data* ofereça suporte para o gerenciamento de APIs, porque este é o meio no qual a plataforma de dados é disponibilizada para os usuários. Na arquitetura proposta, o componente de Integração de Dados é a solução técnica para a criação de serviços de dados, acessando bancos NoSQL ou o *cluster* Hadoop. Para atender a esta demanda, o servidor de API é permanente e as VMs não são liberadas, apenas redimensionadas no caso de picos de processamento.

4.4.7 Sistema de Mensageria

Os sistemas de mensageria são amplamente usados para processamento em tempo real, pois permitem a gravação de registros em alta velocidade. Assim, um sistema de mensageria pode ser usado para fazer a integração entre as fontes de dados existentes e o sistema de *big data*, considerando a quantidade de eventos gerados pelos usuários, a complexidade desses dados e a escalabilidade para suportar as oscilações na demanda dos sistemas [26, 85].

Desta forma, os cenários para utilização de sistemas de mensageria são: (i) fluxo contínuo de dados, quando as fontes de dados produzem novos registros a todo momento; (ii) fluxo desordenado de dados, quando os registros não chegam no destino em uma sequência ordenada; e (iii) fluxo de dados sem limite de tamanho, de forma que não é possível prever nem a natureza nem a complexidade dos dados [98].

Na arquitetura proposta, o sistema de mensageria é usado como uma opção escalável para inclusão de dados. As outras opções de inserção de novos registros são por meio

do componente de Integração de Dados e pela ferramenta de ETL de *big data*. Assim, são oferecidas diversas possibilidades como forma de flexibilizar a operação de inserção de dados, pois a arquitetura deve suportar diversas tecnologias, fontes, volumes e frequências diferentes de gravação de registros.

4.5 Avaliação da Arquitetura

A avaliação da arquitetura proposta foi feita com a criação de uma prova de conceito, do inglês *proof-of-concept* (PoC), na qual foram verificados os cenários de uso e o comportamento do sistema. Desta forma, foi possível determinar os pontos positivos e os negativos do projeto. Após a definição das funcionalidades do BDaaS, foram conduzidos experimentos utilizando as técnicas e as ferramentas para criação dos sistemas de *big data* para se chegar à combinação mais adequada.

O projeto *Big Data Access Tool* (BDAT) [88] é a implementação prática da PoC e foi usado para avaliar as capacidades da arquitetura proposta na forma de um sistema de *big data*. O BDAT foi escrito em Java e incorpora os *frameworks* para *big data*, as rotinas de ETL, o gerenciamento da API e o sistema de mensageria. Considerando a diversidade de tecnologias disponíveis na área de *big data*, o BDAT representa uma camada de abstração entre as funcionalidades de um sistema de *big data* e os respectivos softwares de implementação, podendo ser usado para criar novos sistemas de *big data*.

O *dataset* é composto por diversas tabelas de sistemas jurídicos disponíveis no TJDFT, totalizando aproximadamente 1,5 bilhão de registros que foram importados do SGBDR corporativo. O *cluster* Hadoop/Spark usado nos experimentos tem quatro nós, um *master* e três *worker nodes*, conforme mostrado na Figura 4.5. As avaliações foram realizadas com a simulação de atividades rotineiras, como a execução de comandos SQL para extração de informações do *dataset*, tais como as mostradas na Tabela 4.3. No SGBDR os campos utilizados para consolidação de dados são indexados e particionados, em um alto nível de otimização. No *cluster* foram usadas ferramentas de análise em arquivos gravados no HDFS e no *object storage*.

A arquitetura proposta nesta dissertação envolve as áreas de *Big Data*, de Computação em Nuvem e a intersecção entre elas. Por isso, foi avaliada sob diferentes perspectivas. Inicialmente, ela foi avaliada como uma arquitetura de referência independente das tecnologias e das implementações, de forma a contribuir para a pesquisa e o desenvolvimento dos sistemas de *big data*. Por fim, são classificados os serviços, as tecnologias e como se relacionam com o ambiente de nuvem privada.

4.5.1 Roteiro de Implantação

A implantação da nuvem privada foi feita com o OpenStack [104] e com seus módulos específicos que adicionam suporte para *big data* [101] e bancos de dados [106]. A instalação está detalhada no Anexo I, no qual estão descritos os principais procedimentos e os parâmetros para configuração da plataforma de nuvem, do *object storage*, da rede, do *cluster* Hadoop e das imagens para as VMs. O Anexo II mostra as rotinas de ETL utilizadas no experimento e o Anexo III mostra um exemplo de análise de dados com o Spark.

Além da interface Web, o OpenStack oferece a opção de operação por meio de linha de comando, que foi a opção usada nesta dissertação. Após a configuração completa do ambiente, é possível provisionar um *cluster* Hadoop com um único comando, que está disponível no Anexo I. Com a plataforma de nuvem operacional, o próximo passo foi o provisionamento dos serviços. Assim, o roteiro usado para a criação da PoC (Seção 4.5.2) e para a carga de dados inicial usando o BDAT na arquitetura proposta consiste dos seguintes passos:

1. Provisionar o *cluster* de *big data* (Anexo I);
2. Provisionar uma instância do Servidor NoSQL (Cassandra);
3. Provisionar uma instância do Servidor de API (Spring Boot), com a aplicação disponível em [88];
4. Listar as tabelas disponíveis do ambiente do SGBDR (Anexo II);
5. Importar cada tabela para a área de *staging* do sistema de *big data* (Anexo II);
6. Converter os arquivos importados para o formato Avro [15] e gravá-los no HDFS e no *data lake* (Anexo **??**);
7. Criar as tabelas no Cassandra e carregá-las com os arquivos importados com a API Web da Figura 4.10;
8. Realizar as análises no *dataset* com o *cluster* Hadoop e Spark, disponíveis no Anexo III;
9. Realizar as consultas por meio da API Web, disponível na Figura 4.10;
10. Liberar o *cluster*.

4.5.2 Prova de Conceito

O sistema escolhido para o experimento foi o Relatório de Metas Nacionais do Poder Judiciário [36], que é uma análise da produtividade de cada tribunal do país. Neste sis-

tema foram simuladas as demandas de carga de trabalho e o provisionamento de recursos computacionais.

O primeiro experimento foi a análise do *dataset* completo com as ferramentas de processamento em *batch* e de tempo real. As análises foram realizadas com a execução de comandos SQL no *cluster* Hadoop/Spark e no SGBDR. O segundo experimento foi a disponibilização dos registros no banco de dados NoSQL. Por fim, a última situação simulada foi o consumo dos registros por meio da API Web [24, 131]. As operações disponíveis na API Web podem ser vistas na Figura 4.10 e estão divididas em três categorias:

- *Data Access*: operações de inclusão, alteração e consulta dos dados;
- *Data Store*: lista as tabelas disponíveis na PoC;
- ETL: operações de importação e exportação dos dados, bem como lista das tabelas disponíveis no SGBDR.

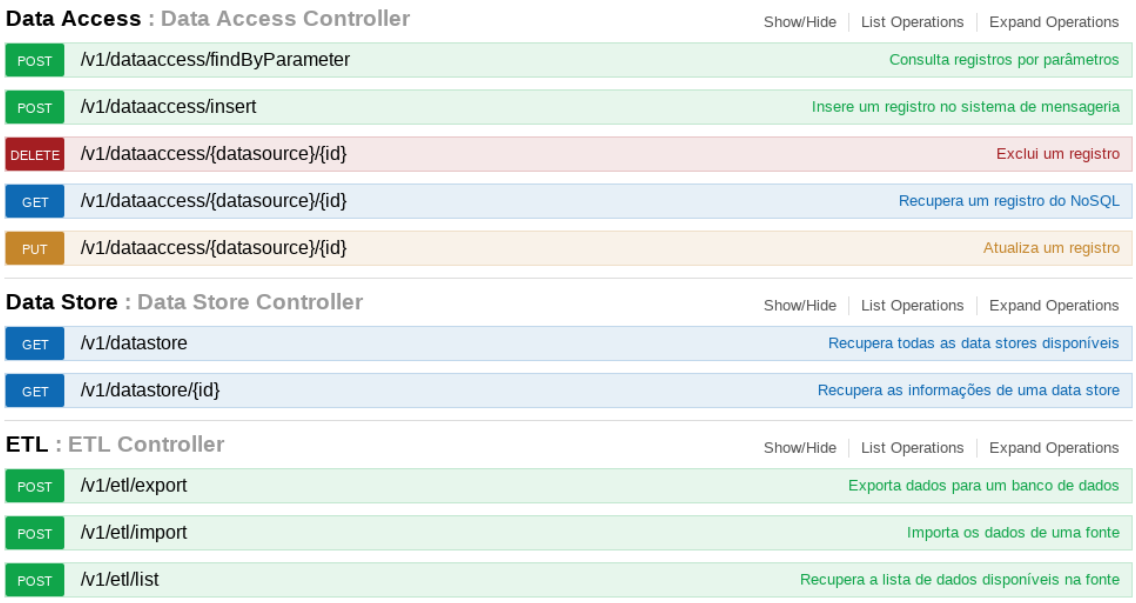

Figura 4.10: API Web com as operações disponíveis no sistema.

Em cada experimento foram realizados testes com três níveis de carga: (i) leve, com até 10 milhões de registros; (ii) moderada, com até 100 milhões de registros; e (iii) alta, a partir de 100 milhões de registros. Assim, foi verificada a quantidade mínima de recursos necessários para suportar os experimentos, evitando o superdimensionamento ou o subdimensionamento dos recursos.

A construção da prova de conceito permitiu verificar que os sistemas de *big data* que usam a arquitetura proposta apresentam as características esperadas no modelo de *Big Data as a Service*, como pode ser conferido na Tabela 4.2.

Além dessas características, foram detectados erros (*bugs*) e deficiências no módulo de *big data* do OpenStack (Sahara). O primeiro *bug* foi encontrado no provisionamento de *cluster* Cloudera, o qual mostra uma mensagem de erro na configuração do HDFS. Contudo, é um *bug* que não prejudica o funcionamento do sistema. O segundo *bug* foi detectado na rotina para alterar o número de nós do *cluster*. Se a rotina for disparada diversas vezes, aumentando e diminuindo o número de nós, pode ocorrer um erro no provisionamento e o *cluster* fica comprometido. Neste caso deve-se criar um novo *cluster*. Uma das deficiências é que os serviços do Hadoop não são reiniciados junto com a VM, de forma que devem ser inicializados manualmente neste caso. A outra deficiência está relacionada com a geração das imagens para os serviços de *big data*, um procedimento complexo e demorado que está descrito em [102] e [103].

4.6 Resultados

Nesta seção estão detalhados os resultados coletados dos experimentos, considerando tanto o ambiente de nuvem privada quanto o sistema de *big data*. Assim, os resultados são analisados a partir das funcionalidades alcançadas pelo sistema construído a partir da arquitetura, ou seja, os requisitos funcionais, e o desempenho na execução de tarefas práticas com os dados reais, ou seja, os requisitos de performance.

4.6.1 Métricas para Avaliação

As métricas usadas para a avaliação examinam aspectos relacionados com as funcionalidade e com a performance. Assim, as características que se espera alcançar com os sistemas criados nesta arquitetura são:

- Conformidade com a definição de *Big Data as a Service*: a arquitetura oferece os serviços definidos na Seção 2.3.2, que são o DaaS, o DBaaS, o AaaS e o StaaS;
- Adequação ao ambiente de nuvem privada: a arquitetura pode ser operada em ambiente de nuvem, preferencialmente por linha de comandos;
- Conformidade com arquiteturas de referência consolidadas, como [23, 26, 89]: a arquitetura proposta se baseia na estrutura das referências, entretanto, a implantação deve ser feita unicamente com tecnologias de *big data*, evitando as alternativas tradicionais como SGBDR e *storages*;
- Performance do sistema: o tempo de resposta das consultas por registros individuais é da ordem de centésimos de segundo, a resposta das análises nos *datasets* é da ordem de minutos e o espaço em disco é comparável ao *datacenter* tradicional.

Tabela 4.2: Comparativo com a Arquitetura Proposta.

	Característica										
Trabalho	*Batch*	RT	Hadoop	NoSQL	Cloud	Private	BDaaS	API	PoC	*Ad Hoc*	Referência
Gao [55]	X	X							X	X	
Zheng *et al.* [131]							X	X	X	X	
Marz e Warren [89]	X	X			X			X	X		
Jambi [68]	X	X	X	X	X	X	X	X	X		
Simmhan *et al.* [116]	X	X	X		X	X			X		
Eftekhari *et al.* [46]	X	X	X	X	X	X	X	X	X		
Horey *et al.* [63]	X	X	X		X						
Thaha *et al.* [120]	X	X	X		X	X		X	X		
Chang [25]	X	X			X	X			X		
Xu *et al.* [129]	X				X	X		X			
Adnan *et al.* [2]	X		X		X	X			X		
Conejero *et al.* [34]	X		X		X	X					
Vogel *et al.* [123]	X		X		X	X			X		
Corradi *et al.* [39]	X		X		X	X			X		
Maier [86]	X	X	X	X	X						X
Hu *et al.* [65]	X	X	X	X	X			X			X
Chang [26]	X	X		X	X	X			X	X	X
Bhagattjee [23]	X	X	X	X	X		X	X		X	X
Assunção *et al.* [21]	X	X	X	X	X	X	X	X		X	X
Arquitetura proposta	X	X	X	X	X	X	X	X	X	X	X

4.6.2 Requisitos Funcionais

Com relação às funcionalidades, os experimentos concluíram que a arquitetura proposta atende às características essenciais da computação em nuvem definidas em [90], combinadas com as demandas dos sistemas de *big data*, as quais são:

- Auto-serviço por demanda: provisionamento e configuração de *clusters* Hadoop/Spark e de NoSQL (Cassandra) de forma automatizada;
- Amplo acesso à rede: os serviços de provisionamento do BDaaS e também os serviços de dados estão disponíveis na rede usando mecanismos padronizados;
- Agrupamento de recursos: os recursos computacionais são compartilhados entre os consumidores em um modelo multiusuário (*multi-tenant*);
- Elasticidade rápida: característica essencial do BDaaS, a arquitetura suporta a adição ou remoção de novos nós ao *cluster* de forma automatizada;
- Serviço mensurável: apesar de não haver cobrança financeira, a arquitetura permite a medição do uso dos recursos com a combinação dos módulos de monitoramento da plataforma de nuvem e do *cluster*.

A arquitetura proposta suporta esses requisitos por meio da combinação de técnicas e de ferramentas para ajudar a resolver dificuldades enfrentadas durante a criação e a implantação dos sistemas de *big data*. É o caso do auto-serviço por demanda, que diminuiu o tempo de instalação e configuração do *cluster* de horas para minutos. Assim, permite que os sistemas de *big data* sejam implantados mais rapidamente.

A utilização de padrões abertos e de *web services* garantiu a facilidade de operação da infraestrutura e a publicação dos serviços na Internet. Essa característica é importante por causa da velocidade com que surgem novas tecnologias, garantindo durabilidade para a arquitetura proposta, uma vez que é possível adicionar novas ferramentas facilmente.

O agrupamento de recurso se aplica ao armazenamento, processamento, memória e rede, que já fazem parte das características da computação em nuvem pública. Além destas, a arquitetura proposta acrescentou os serviços de dados, por meio do *data lake*, uma tecnologia recente que permite um nível elevado de desacoplamento entre os dados e os sistemas.

A elasticidade rápida permitiu que os recursos fossem redimensionados, aumentando ou diminuindo a quantidade de nós do *cluster*. Essa característica foi usada para adequar o poder de processamento à demanda do usuário. Por fim, o requisito do serviço mensurável foi usado para verificar a real necessidade de hardware, com a finalidade de observar a ociosidade do sistema.

4.6.3 Requisitos de Performance

Com relação à performance, o objetivo é que a PoC tenha desempenho similar ao verificado no *datacenter* tradicional. Dessa forma, foram usados dois conjuntos de equipamentos. A plataforma de nuvem privada usada na PoC usou quatro servidores Dell PowerEdge R710 Intel Xeon X5560 2.80 GHz, cada um 8 processadores, 96 GB de memória e 256 GB de disco. O SGBDR usa um servidor Blade Intel Xeon 2.4 GHZ com 16 processadores, 64 GB de memória e 7 TB de disco.

Os resultados obtidos nos experimentos estão listados na Tabela 4.3, na qual estãos relacionados o nome da operação realizada, a carga de trabalho aplicada e a quantidade de nós utilizada para o processamento. Cada operação foi realizada ao menos dez vezes, sendo que estão mostrados os tempos médios em segundos.

Tabela 4.3: Resultados Medidos em Segundos.

Item	**Carga leve**	**Carga moderada**	**Carga alta**
Provisionamento do *cluster*	160 (2 nós)	180 (3 nós)	190 (4 nós)
Tamanho do *dataset* (Avro)	400 MB	4 GB	18 GB
Tamanho do *dataset* no SGBDR	-	-	76 GB
Análise no *cluster*	8	80	150
Análise no SGBDR	6	90	201
Consulta registro no NoSQL	0,03	0,06	0,1
Consulta registro no SGBDR	0,01	0,02	0,04

O primeiro item da tabela é o provisionamento do *cluster* Hadoop de acordo com cada carga de trabalho descrita na Seção 4.5.2. O *cluster* é criado por demanda e seus recursos são liberados tão logo as análises sejam concluídas. Foi verificado que não há uma grande diferença de tempo no provisionamento com dois, três ou quatro nós. Esse fato pode ser explicado porque as VMs são criadas em paralelo, de forma que o software é instalado simultaneamente em todas elas.

O segundo item é o tamanho do *dataset* em formato Avro e no SGBDR. Na PoC os dados foram gravados no *object storage*, enquanto que o SGBDR utiliza o *storage* corporativo. A partir destes *datasets* foram executadas consultas SQL idênticas em cada ambiente, tanto na PoC quanto no SGBDR. Os tempos médios medidos nas análises estão no terceiro item da tabela.

Por fim, o último item mostra a consulta por registros individuais, baseando-se no valor da chave-primária da tabela em cada uma das cargas de trabalho (leve, moderada e alta). No SGBDR as consultas foram feitas utilizando o Squirrel[5], que é um cliente de SQL. Na PoC as medidas foram feitas na API Web por meio do Postman[6], que é um

[5]http://squirrel-sql.sourceforge.net/
[6]https://www.getpostman.com/

cliente para aplicações REST. A consulta no NoSQL não se mostrou tão rápida quanto a do SGBDR, entretanto, apresenta performance aceitável para os padrões dos sistemas corporativos do TJDFT, observando que a resposta está perto de 0,1 segundo. Uma maior otimização pode ser feita com o estudo detalhado dos *datasets* e a utlização de técnicas como o particionamento [29].

O estudo do estado da arte também permitiu concluir que é mais interessante o armazenamento dos dados no *object storage* do que no HDFS, uma vez que não há grande diferença de performance entre as tecnologias. Este ponto reforça a importância do baixo acomplamento na arquitetura proposta, e é apontado como uma tendência, juntamente com o avanço dos *data lakes*.

Os resultados da Tabela 4.3 permitem concluir que a correta combinação de tecnologias e técnicas de construção de sistemas de *big data* na nuvem garantem um desempenho compatível com o *datacenter* tradicional, considerando as métricas de tempo de resposta e espaço em disco. Um ponto importante a ser listado é a performance e a economia de disco possibilitado pela compactação de dados dos novos formatos de arquivos, como o Avro.

4.7 Considerações Finais

Neste capítulo foi apresentada a arquitetura para sistemas de *big data* como um serviço em nuvem privada. Para isso, foram detalhadas as camadas e as funcionalidades, bem como os serviços que serão oferecidos, com o objetivo de atender as necessidades das organização na implantação deste tipo de solução.

Os serviços ofertados pela plataforma abstraem a complexidade da implementação e da configuração das tecnologias subjacentes por meio dos recursos de nuvem computacional. Com isso, os usuários têm acesso imediato aos dados e aos mecanismos de análise. Esses serviços incluem a criação de *cluster* de *big data*, gerenciamento, análise e armazenamento de dados. Com isso, a arquitetura suporta os modos de processamento *batch* e tempo real, armazenamento distribuído, banco de dados NoSQL e API de *web services* para exposição de dados na rede.

As métricas de monitoramento possibilitam o aperfeiçoamento da elasticidade e escalabilidade, bem como a quantidade adequada de recursos para cada carga de trabalho. Isso é possível porque os recursos são ajustados de acordo com a demanda, aumentando ou diminuindo a quantidade de nós. Como consequência, essa elasticidade ajuda a evitar o superdimensionamento e o subdimensionamento, ambos prejudiciais à qualidade do serviço.

Os resultados dos experimentos comprovaram que a performance da arquitetura proposta é adequada para a sua utilização em ambiente corporativo, uma vez que o desempenho é comparável ao do *datacenter* tradicional, entretanto, a arquitetura apresenta vantagens do custo reduzido e operação automatizada. O menor custo é obtido porque é possível aumentar o volume de dados sem a necessidade de hardware de alto custo, enquanto que a operação automatizada diminui a necessidade de intervenção manual da equipe de infraestrutura.

Capítulo 5

Conclusão

O modelo de BDaaS está disponível nos principais provedores de nuvem pública, entretanto, é pouco explorado em nuvens privadas, o que oferece oportunidades para pesquisa e aperfeiçoamento das soluções existentes. Desta forma, esta área de pesquisa, que une o *big data* e a computação em nuvem, tem atraído atenção da indústria e da academia.

Dentro deste contexto, este trabalho fez uma proposta de arquitetura para *Big Data as a Service* em nuvem privada, que funciona como referência durante a adoção destas tecnologias. A infraestrutura compreende desde as camadas mais básicas de hardware, passando pelo gerenciamento dos dados e terminando com a oferta de serviços de alto nível para análise de dados e integração com sistemas pré-existentes na organização. Com esta descrição, em um curto espaço de tempo, é possível entregar soluções de *big data* sem passar por fases como pesquisa e teste de ferramentas.

Assim, este estudo descreveu as funcionalidades e as propostas de soluções para a área de *big data* em nuvem privada, adicionando casos de uso práticos avaliados em cenários reais com dados do Governo Federal. Como resultado, foi formalizada uma descrição arquitetural com as técnicas específicas de criação de sistemas, na forma de um *roadmap* de tecnologias que pode ser usado para implantar novas soluções, ou como ferramenta de comunicação com usuários não técnicos.

As características dos sistemas construídos a partir da arquitetura proposta e verificados no estudo de caso incluem alta performance, escalabilidade, modularidade, baixo acoplamento, reusabilidade da arquitetura e uso de padrões abertos, sendo estes os pontos positivos verificados. Desta forma, as contribuições desta dissertação são:

- A formalização da arquitetura de *Big Data as a Service* (BDaaS) em nuvem privada para provimento de serviços de dados, com o uso exclusivamente de ferramentas de *big data*;

- A definição de um modelo conceitual para comunicação com os *stakeholders* não técnicos evolvidos no projeto, incluindo uma clara conceituação do modelo de BDaaS;
- A implementação de um *framework* especializado em sistemas de *big data*, para importação de registros e disponibilização na forma de *web services*;
- Um *roadmap* de técnicas, tecnologias e softwares para implantação efetiva de sistemas de *big data*, com o mapeamento entre as funcionalidades desejadas e a forma de sua implementação prática;
- As técnicas para criação da plataforma de computação em nuvem que permitem a instalação e a configuração de *clusters* de *big data* de forma automatizada.

5.1 Trabalhos Futuros

O estado da arte da área de *big data* e da computação em nuvem apresentam diversos desafios ainda abertos, que foram identificados durante a revisão bibliográfica e o desenvolvimento da prova de conceito.

Uma vez que a proposta descrita nesta dissertação tem escopo e restrições bem definidas, não foram incluídas várias possibilidades de uso, como ferramentas e funcionalidades para tarefas mais específicas. Assim, as possíveis evoluções para a arquitetura proposta nesta trabalho incluem:

- A orquestração dos serviços com Kubernetes [79] e em *containers*, como Docker [1] e LXC [83];
- Evolução do mecanismo de balanceamento de carga no disco, considerando o desequilíbrio entre as capacidades de CPU e de I/O;
- Disponibilizar um modelo de segurança e de compartilhamento de dados, considerando uma nuvem multi-usuário;
- Metodologia de pré-processamento de *big data* para garantir qualidade e limpeza dos dados;

Ainda como trabalho futuro, a própria arquitetura proposta pode ter módulos automatizados para a otimização dos serviços de dados, aplicados na criação do *data lake* e das tabelas NoSQL. Neste caso, seriam usadas técnicas de *machine learning* para descobrir a forma mais performática de criar esses serviços, por meio da combinação das diversas tecnologias disponíveis.

Referências

[1] Docker. https://www.docker.com/, 2018. 76

[2] Muhammad Adnan, Muhammad Afzal, Muhammad Aslam, Roohi Jan, *and* AM Martinez-Enriquez. Minimizing big data problems using cloud computing based on hadoop architecture. In *High-capacity Optical Networks and Emerging/Enabling Technologies (HONET), 2014 11th Annual*, pages 99–103. IEEE, 2014. 37, 44, 70

[3] Amazon. Amazon EC2. https://aws.amazon.com/ec2/, 2018. 6, 14

[4] Amazon. Amazon S3. https://aws.amazon.com/s3/, 2018. 14, 59, 62

[5] Apache Software Foundation. Apache Cassandra. http://cassandra.apache.org/, 2017. 21, 27, 58

[6] Apache Software Foundation. Apache Hadoop. http://hadoop.apache.org/, 2017. 2, 5, 17, 22, 25, 35, 49

[7] Apache Software Foundation. Hadoop Common. https://github.com/apache/hadoop-common, 2017. 25

[8] Apache Software Foundation. Apache HBase. https://hbase.apache.org/, 2017. 21, 27, 34, 36

[9] Apache Software Foundation. HDFS. http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html, 2017. 21, 26

[10] Apache Software Foundation. Apache Hive. https://hive.apache.org/, 2017. 25, 27, 57

[11] Apache Software Foundation. MapReduce. http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html, 2017. 22, 23, 25

[12] Apache Software Foundation. Apache Spark. https://spark.apache.org/, 2017. 25, 49

[13] Apache Software Foundation. Apache Tez. https://tez.apache.org, 2017. 25

[14] Apache Software Foundation. YARN. http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html, 2017. 26

[15] Apache Software Foundation. Avro. https://avro.apache.org/, 2018. 63, 67

[16] Apache Software Foundation. Kafka. `https://kafka.apache.org/`, 2018. 57, 58

[17] Apache Software Foundation. Parquet. `https://parquet.apache.org/`, 2018. 63

[18] Apache Software Foundation. Apache Pig. `https://pig.apache.org/`, 2018. 25

[19] Raja Appuswamy, Christos Gkantsidis, Dushyanth Narayanan, Orion Hodson, *and* Antony Rowstron. Scale-up vs scale-out for hadoop: Time to rethink? In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 20. ACM, 2013. 24, 61

[20] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, *et al.* A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010. 2, 9, 10, 12, 13

[21] Marcos D Assunção, Rodrigo N Calheiros, Silvia Bianchi, Marco AS Netto, *and* Rajkumar Buyya. Big Data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, 79:3–15, 2015. 12, 41, 44, 60, 70

[22] AWS. API Gateway. `https://aws.amazon.com/api-gateway/`, 2018. 65

[23] Benoy Bhagattjee. *Emergence and taxonomy of Big Data as a service.* PhD thesis, Massachusetts Institute of Technology, 2014. 2, 20, 21, 22, 40, 41, 44, 47, 69, 70

[24] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, *and* Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009. 6, 68

[25] Victor Chang. Towards a Big Data system disaster recovery in a Private Cloud. *Ad Hoc Networks*, 35:65–82, 2015. 36, 44, 70

[26] Wo L. Chang. Big Data Interoperability Framework: Volume 6, Reference Architecture. *NIST special publication, Information Technology Laboratory, Gaithersburg*, 6, 2015. xi, 21, 39, 40, 41, 44, 47, 56, 65, 69, 70

[27] Wo L. Chang. Big Data Interoperability Framework: Volume 1, Definitions. *NIST special publication, Information Technology Laboratory, Gaithersburg*, 1, 2015. 2, 16, 18, 27

[28] Wo L Chang. NIST Big Data Interoperability Framework: Volume 5, Architectures White Paper Survey. Technical report, 2015. 47

[29] Artem Chebotko, Andrey Kashlev, *and* Shiyong Lu. A Big Data Modeling Methodology for Apache Cassandra. In *Big Data (BigData Congress), 2015 IEEE International Congress on*, pages 238–245. IEEE, 2015. 64, 73

[30] CL Philip Chen *and* Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences*, 275: 314–347, 2014. 46, 48

[31] Min Chen, Shiwen Mao, *and* Yunhao Liu. Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209, 2014. 27

[32] Cloudera. Cloudera. `https://www.cloudera.com`, 2018. 49

[33] CloudStack. CloudStack. `https://cloudstack.apache.org/`, 2018. 12, 14, 38

[34] Javier Conejero, Blanca Caminero, *and* Carmen Carrión. Analysing hadoop performance in a multi-user iaas cloud. In *High Performance Computing & Simulation (HPCS), 2014 International Conference on*, pages 399–406. IEEE, 2014. 37, 44, 70

[35] Conselho Nacional de Justiça. Termo de Cooperação Técnica N. 058/2009. `http://www.cnj.jus.br/images/dti/Comite_Gestao_TIC/Modelo_Nacional_Interoperabilidade/tcot_n_58_2009.pdf`, 2009. 1

[36] Conselho Nacional de Justiça. Metas Nacionais. `http://www.cnj.jus.br/gestao-e-planejamento/metas`, 2009. 67

[37] Conselho Nacional de Justiça. Resolução conjunta nº 3 de 16/04/2013. `http://www.cnj.jus.br/busca-atos-adm?documento=229`, 2013. 1

[38] Conselho Nacional de Justiça. Resolução nº 194 de 26/05/2014. `http://www.cnj.jus.br/busca-atos-adm?documento=2483`, 2014. 1, 2

[39] Antonio Corradi, Luca Foschini, Valerio Pipolo, *and* Alessandro Pernafini. Elastic provisioning of virtual hadoop clusters in openstack-based clouds. In *Communication Workshop (ICCW), 2015 IEEE International Conference on*, pages 1914–1920. IEEE, 2015. 38, 44, 59, 60, 70

[40] Breno G. S. Costa, Marco Antonio Sousa Reis, Aletéia P. F. Araújo, *and* Priscila Solis. Performance and Cost Analysis Between On-Demand and Preemptive Virtual Machines. In *Proceedings of the 8th International Conference on Cloud Computing and Services Science, CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018.*, pages 169–178. SciTePress, 2018. doi: 10.5220/0006709001690178. URL `https://doi.org/10.5220/0006709001690178`. 3

[41] Michael Cox *and* David Ellsworth. Application-controlled demand paging for out-of-core visualization. In *Proceedings of the 8th conference on Visualization'97*, pages 235–ff. IEEE Computer Society Press, 1997. 2

[42] Jeffrey Dean *and* Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. 23, 24

[43] James Dixon. Pentaho, Hadoop, and Data Lakes. `https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/`, 2010. 63

[44] Robert Dukaric *and* Matjaz B Juric. Towards a unified taxonomy and architecture of cloud frameworks. *Future Generation Computer Systems*, 29(5):1196–1210, 2013. 11, 47

[45] eBay. Developer Program. `https://go.developer.ebay.com/`, 2018. 65

[46] Azadeh Eftekhari, Farhana Zulkernine, *and* Patrick Martin. Binary: A framework for big data integration for ad-hoc querying. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 2746–2753. IEEE, 2016. xi, 34, 35, 44, 70

[47] EMC. Big Data and Big Data Analytics. `https://www.dellemc.com/en-us/big-data/index.htm`, 2018. 46

[48] Eucalyptus. Cloud-computing Platform. `https://github.com/eucalyptus/eucalyptus`, 2018. 12

[49] Michael Factor, Kalman Meth, Dalit Naor, Ohad Rodeh, *and* Julian Satran. Object storage: The future building block for storage systems. In *Local to Global Data Interoperability-Challenges and Technologies, 2005*, pages 119–123. IEEE, 2005. 62

[50] Huang Fang. Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE International Conference on*, pages 820–824. IEEE, 2015. 63

[51] Heitor Faria, Rodrigo Hagstrom, Marco Reis, Breno G. S. Costa, Edward de Oliveira Ribeiro, Maristela Holanda, Priscila Solis Barreto, *and* Aletéia P. F. Araújo. A Hadoop Open Source Backup Solution. In *Proceedings of the 8th International Conference on Cloud Computing and Services Science, CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018.*, pages 651–657. SciTePress, 2018. doi: 10.5220/0006809206510657. URL `https://doi.org/10.5220/0006809206510657`. 3

[52] Roy T Fielding *and* Richard N Taylor. *Architectural styles and the design of network-based software architectures*. University of California, Irvine Doctoral dissertation, 2000. 65

[53] Ian Foster, Yong Zhao, Ioan Raicu, *and* Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. Ieee, 2008. 6

[54] John Gantz *and* David Reinsel. Extracting value from chaos. *IDC iview*, 1142 (2011):1–12, 2011. 17

[55] Xiaoming Gao. *Scalable architecture for integrated batch and streaming analysis of big data*. PhD thesis, Indiana University, 2015. xi, 30, 31, 44, 70

[56] Gartner. Welcome to the API Economy. `https://www.gartner.com/smarterwithgartner/welcome-to-the-api-economy/`, 2018. 65

[57] Tatiana Engel Gerhardt *and* Denise Tolfo Silveira. *Métodos de pesquisa*. Plageder, 2009. 4

[58] Google. API Explorer. `https://developers.google.com/apis-explorer/`, 2018. 65

[59] Google. Cloud Storage. `https://cloud.google.com/storage/docs/`, 2018. 62

[60] Governo Federal. LAI: A Lei de Acesso à Informação. http://www.acessoainformacao.gov.br/assuntos/conheca-seu-direito/a-lei-de-acesso-a-informacao, 2011. 1

[61] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, *and* Samee Ullah Khan. The rise of "Big Data"on cloud computing: Review and open research issues. *Information Systems*, 47:98–115, 2015. xi, 13, 17, 18, 21, 22, 25, 27, 46

[62] Brian Hopkins, Boris Evelson, B Hopkins, B Evelson, S Leaver, C Moore, A Cullen, M Gilpin, *and* M Cahill. Expand your digital horizon with big data. *http://www.asterdata.com/newsletter-images/30-04-2012/resources/forrester_expand_your_digital_horiz.pdf*, 2011. xi, 18, 46

[63] James L Horey, Edmon Begoli, Raghul Gunasekaran, Seung-Hwan Lim, *and* James J Nutaro. Big data platforms as a service: Challenges and approach. In *HotCloud*, 2012. 13, 20, 35, 36, 44, 70

[64] HortonWorks. HortonWorks. https://hortonworks.com/, 2017. 49

[65] Han Hu, Yonggang Wen, Tat-Seng Chua, *and* Xuelong Li. Toward scalable systems for big data analytics: A technology tutorial. *IEEE access*, 2:652–687, 2014. xi, 19, 38, 39, 44, 70

[66] ISO. Systems and software engineering–architecture description. Technical report, ISO/IEC/IEEE 42010, 2011. 4

[67] Yashpalsinh Jadeja *and* Kirit Modi. Cloud computing-concepts, architecture and challenges. In *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, pages 877–880. IEEE, 2012. 13

[68] Sahar Hussain Jambi. *Engineering Scalable Distributed Services for Real-Time Big Data Analytics*. PhD thesis, University of Colorado at Boulder, 2016. xi, 33, 34, 44, 70

[69] Pooyan Jamshidi, Claus Pahl, Nabor C Mendonça, James Lewis, *and* Stefan Tilkov. Microservices: The journey so far and challenges ahead. *IEEE Software*, 35(3):24–35, 2018. 57

[70] JSON. JavaScript Object Notation. https://www.json.org/, 2018. 65

[71] João Bachiega Junior, Marco Antonio Sousa Reis, Aletéia P. F. Araújo, *and* Maristela Holanda. Cost Analysis for Big Geospatial Data Processing in Public Cloud Providers. In *Cloud Computing and Service Science - 7th International Conference, CLOSER 2017, Porto, Portugal, April 24-26, 2017, Revised Selected Papers*, volume 864 of *Communications in Computer and Information Science*, pages 223–236. Springer, 2017. doi: 10.1007/978-3-319-94959-8_12. URL https://doi.org/10.1007/978-3-319-94959-8_12. 3

[72] João Bachiega Junior, Marco Antonio Sousa Reis, Aletéia Patrícia Favacho de Araújo, *and* Maristela Holanda. Cost Optimization on Public Cloud Provider for Big Geospatial Data. In *CLOSER 2017 - Proceedings of the 7th International Conference on Cloud Computing and Services Science, Porto, Portugal, April 24-26, 2017.*, pages 54–62. SciTePress, 2017. doi: 10.5220/0006237800540062. URL `https://doi.org/10.5220/0006237800540062`. 3

[73] João Bachiega Junior, Marco Antonio Sousa Reis, Maristela Holanda, *and* Aleteia P. F. Araujo. Comparacao de desempenho na indexacao de big geospatial Data em ambiente de nuvem computacional. In *XVIII Brazilian Symposium on Geoinformatics - GeoInfo 2017, Salvador, BA, Brazil, December 04-06, 2017*, pages 134–139. MCTIC/INPE, 2017. URL `http://urlib.net/8JMKD3MGPDW34P/3Q5DTTP`. 3

[74] João Bachiega Junior, Marco Antonio Sousa Reis, Aletéia Patrícia Favacho de Araújo, *and* Maristela Holanda. A Cost-Efficient Method for Big Geospatial Data on Public Cloud Providers (GEOProcessing 2017. In *GEOProcessing 2017 : The Ninth International Conference on Advanced Geographic Information Systems, Applications, and Services*. IARIA, 2017. 3

[75] João Bachiega Junior, Marco Antonio Sousa Reis, Maristela Holanda, *and* Aletéia Patrícia Favacho de Araújo. An Architecture for Cost Optimization in the Processing of Big Geospatial Data in Public Cloud Providers. In *Big Data (BigData Congress): Workshop, 2018 IEEE International Congress on*. IEEE, 2018. 3

[76] Karthik Kambatla, Giorgos Kollias, Vipin Kumar, *and* Ananth Grama. Trends in Big Data analytics. *Journal of Parallel and Distributed Computing*, 74(7):2561–2573, 2014. 10, 58

[77] Lisa Kart, Nick Heudecker, *and* Frank Buytendijk. Survey analysis: big data adoption in 2013 shows substance behind the hype. *Gartner Report GG0255160*, 2013. 13

[78] Avita Katal, Mohammad Wazid, *and* RH Goudar. Big data: issues, challenges, tools and good practices. In *Contemporary Computing (IC3), 2013 Sixth International Conference on*, pages 404–409. IEEE, 2013. 46

[79] Kubernetes. Production-Grade Container Orchestration. `https://kubernetes.io/`, 2018. 76

[80] KVM. Kernel Virtual Machine. `https://www.linux-kvm.org/`, 2018. 14, 15

[81] Chuck Lam. *Hadoop in action*. Manning Publications Co., 2010. 23, 24

[82] Zhuozhao Li, Haiying Shen, Walter Ligon, *and* Jeffrey Denton. An exploration of designing a hybrid scale-up/out hadoop architecture based on performance measurements. *IEEE Transactions on Parallel and Distributed Systems*, 28(2):386–400, 2017. 25

[83] Linux. Linux Containers. `https://linuxcontainers.org/`, 2018. 15, 76

[84] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, *and* Dawn Leaf. *NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology (Special Publication 500-292)*. CreateSpace Independent Publishing Platform, USA, 2012. ISBN 1478168021, 9781478168027. 7, 47

[85] Xiufeng Liu, Nadeem Iftikhar, *and* Xike Xie. Survey of real-time processing systems for big data. In *Proceedings of the 18th International Database Engineering & Applications Symposium*, pages 356–361. ACM, 2014. 63, 65

[86] Markus Maier. Towards a Big Data Reference Architecture. Master's thesis, University of Eindhoven, 2013. 38, 44, 70

[87] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, *and* Angela H Byers. Big data: The next frontier for innovation, competition, and productivity. 2011. 17

[88] Marco Reis. Big Data Access Tool. `https://github.com/masreis/big-data-access-tool`, 2018. 66, 67

[89] Nathan Marz *and* James Warren. *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015. xi, 31, 32, 33, 44, 60, 69, 70

[90] Peter Mell *and* Tim Grance. SP 800-145. The NIST definition of cloud computing. 2011. 1, 7, 8, 12, 71

[91] Microsoft. Big Data and Data Analytics Solutions. `https://www.microsoft.com/en-us/sql-server/big-data`, 2018. 46

[92] Microsoft. Azure Blob Storage. `https://azure.microsoft.com/en-us/services/storage/blobs/`, 2018. 62

[93] Microsoft. Hyper-V. `https://www.microsoft.com/en-us/cloud-platform/server-virtualization`, 2018. 14, 15

[94] Natalia Miloslavskaya *and* Alexander Tolstoy. Big data, fast data and data lake concepts. *Procedia Computer Science*, 88:300–305, 2016. 63

[95] Mirantis. Sahara Data Processing for OpenStack. `https://www.mirantis.com/software/data-processing-sahara/`, 2017. 15

[96] Mirantis. Big data analytics and elastic data processing. `https://www.mirantis.com/solutions/big-data-analytics/`, 2018. 61

[97] MySQL. MySQL. `https://www.mysql.com/`, 2018. 35

[98] Dmitry Namiot. On big data stream processing. *International Journal of Open Information Technologies*, 3(8), 2015. 65

[99] Netflix. OSS. `https://netflix.github.io/`, 2018. 65

[100] OpenNebula. OpenNebula. https://opennebula.org/, 2018. 12, 14, 38

[101] OpenStack. OpenStack Sahara. https://docs.openstack.org/sahara/latest/, 2017. 15, 36, 59, 67

[102] OpenStack. Sahara image elements project. https://github.com/openstack/sahara-image-elements, 2018. 69

[103] OpenStack. Building Guest Images for OpenStack Trove. https://docs.openstack.org/trove/latest/admin/building_guest_images.html, 2018. 69

[104] OpenStack. OpenStack. https://www.openstack.org/, 2018. 5, 12, 14, 36, 38, 59, 67

[105] OpenStack. Swift. https://docs.openstack.org/swift/latest/, 2018. 59

[106] OpenStack. Trove. https://docs.openstack.org/trove/latest/, 2018. 59, 67

[107] Oracle. Oracle Big Data. https://www.oracle.com/big-data/index.html, 2018. 46

[108] Oracle. Oracle Object Storage. https://cloud.oracle.com/storage/object-storage/features, 2018. 62

[109] Oracle VM. Oracle VM. http://www.oracle.com/technetwork/server-storage/vm/overview/index.html, 2018. 14

[110] Cesare Pautasso. Restful web services: principles, patterns, emerging technologies. In *Web Services Foundations*, pages 31–51. Springer, 2014. 56, 65

[111] QEMU. QEMU. http://www.qemu.org/, 2018. 15

[112] R Foundation. The R Project. https://www.r-project.org/, 2017. 34

[113] Nick Rozanski *and* Eóin Woods. *Software systems architecture: working with stakeholders using viewpoints and perspectives*. Addison-Wesley, 2012. 4, 47, 49, 50, 51, 52, 54, 60

[114] Lukas Rupprecht, Rui Zhang, Bill Owen, Peter Pietzuch, *and* Dean Hildebrand. Swiftanalytics: Optimizing object storage for big data analytics. In *Cloud Engineering (IC2E), 2017 IEEE International Conference on*, pages 245–251. IEEE, 2017. 62

[115] Omar Sefraoui, Mohammed Aissaoui, *and* Mohsine Eleuldj. Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3), 2012. 12, 15, 59

[116] Yogesh Simmhan, Saima Aman, Alok Kumbhare, Rongyang Liu, Sam Stevens, Qunzhi Zhou, *and* Viktor Prasanna. Cloud-based software platform for big data analytics in smart grids. *Computing in Science & Engineering*, 15(4):38–47, 2013. 33, 34, 44, 70

[117] Borja Sotomayor, Rubén S Montero, Ignacio M Llorente, *and* Ian Foster. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet computing*, 13(5), 2009. 6, 60

[118] Spring. Spring Boot. `https://spring.io/projects/spring-boot`, 2018. 57

[119] Wei Tan, Yushun Fan, Ahmed Ghoneim, M Anwar Hossain, *and* Schahram Dustdar. From the service-oriented architecture to the web api economy. *IEEE Internet Computing*, 20(4):64–68, 2016. 65

[120] Asmath F Thaha, Manvir Singh, Anang HM Amin, Nazrul M Ahmad, *and* Subarmaniam Kannan. Hadoop in openstack: Data-location-aware cluster provisioning. In *Information and Communication Technologies (WICT), 2014 Fourth World Congress on*, pages 296–301. IEEE, 2014. 36, 44, 60, 70

[121] Muhammad Fahim Uddin, Navarun Gupta, *et al.* Seven V's of Big Data understanding Big Data to extract value. In *American Society for Engineering Education (ASEE Zone 1), 2014 Zone 1 Conference of the*, pages 1–5. IEEE, 2014. 16

[122] VMWare. VMWare. `https://www.vmware.com/`, 2018. 14

[123] Adriano Vogel, Dalvan Griebler, Carlos AF Maron, Claudio Schepke, *and* Luiz Gustavo Fernandes. Private IaaS clouds: a comparative analysis of OpenNebula, CloudStack and OpenStack. In *Parallel, Distributed, and Network-Based Processing (PDP), 2016 24th Euromicro International Conference on*, pages 672–679. IEEE, 2016. 12, 14, 38, 44, 59, 62, 70

[124] Xueying Wang, Zhihui Lu, Jie Wu, Tong Zhao, *and* Patrick Hung. In stechah: An autoscaling scheme for hadoop in the private cloud. In *Services Computing (SCC), 2015 IEEE International Conference on*, pages 395–402. IEEE, 2015. 60

[125] Tom White. *Hadoop: The definitive guide*. "O'Reilly Media, Inc.", 2015. xi, xii, 2, 22, 23, 24, 25, 26

[126] Xen Project. Xen Project. `https://www.xenproject.org/`, 2018. 14, 15

[127] XenServer. XenServer. `https://xenserver.org/`, 2018. 14, 15

[128] E Xinhua, Jing Han, Yasong Wang, *and* Lianru Liu. Big data-as-a-service: Definition and architecture. In *Communication Technology (ICCT), 2013 15th IEEE International Conference on*, pages 738–742. IEEE, 2013. 20

[129] Guanghui Xu, Feng Xu, *and* Hongxu Ma. Deploying and researching hadoop in virtual machines. In *Automation and Logistics (ICAL), 2012 IEEE International Conference on*, pages 395–399. IEEE, 2012. 36, 44, 70

[130] Qi Zhang, Lu Cheng, *and* Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010. 2, 10, 47, 60

[131] Zibin Zheng, Jieming Zhu, *and* Michael R Lyu. Service-generated Big Data and Big Data-as-a-service: an overview. In *Big Data (BigData Congress), 2013 IEEE International Congress on*, pages 403–410. IEEE, 2013. 2, 20, 30, 31, 44, 68, 70

Anexo I

Roteiro de Instalação

```
# Preparação do sistema operacional
getenforce
sed -i 's/enforcing/disabled/g' /etc/selinux/config
setenforce 0
getenforce
yum update -y
#
systemctl disable firewalld
systemctl stop firewalld
systemctl disable NetworkManager
systemctl stop NetworkManager
systemctl enable network
systemctl start network
# OpenStack Pike
yum install -y centos-release-openstack-pike
yum install -y openstack-packstack
packstack --gen-answer-file=/root/answer_file.txt
yum update -y
# Alterar os seguintes parâmetros no arquivo answer_file.txt
% CONFIG_SERVICE_WORKERS=1
% CONFIG_SAHARA_INSTALL=y
% CONFIG_HEAT_INSTALL=y
% CONFIG_TROVE_INSTALL=y
% CONFIG_PROVISION_DEMO=n
% CONFIG_HEAT_CLOUDWATCH_INSTALL=y
% CONFIG_SWIFT_STORAGES=/dev/sdb1
% CONFIG_SWIFT_STORAGE_FSTYPE=xfs
# Instalação do OpenStack
```

```
packstack --answer-file=/root/answer_file.txt
# Configuração da rede
openstack network create external_network \
  --share \
  --project admin \
  --external \
  --provider-network-type flat \
  --provider-physical-network extnet
openstack subnet create external_subnet \
  --allocation-pool start=172.17.74.20,end=172.17.74.40 \
  --no-dhcp \
  --gateway 172.17.72.1 \
  --dns-nameserver 172.16.2.138 \
  --network external_network \
  --subnet-range 172.17.74.0/22
openstack network create net_data_analysis_a
dns=x.x.x.x # Informe o seu DNS
openstack subnet create subnet_data_analysis_a \
  --subnet-range 10.4.4.0/24 \
  --network net_data_analysis_a \
  --dns-nameserver $dns
openstack router create router_data_analysis_a
openstack router add subnet router_data_analysis_a subnet_data_analysis_a
openstack router set \
  --external-gateway external_network router_data_analysis_a
openstack security group create cluster-sec-group --project admin
openstack security group list
sec_group=cluster-sec-group
openstack security group rule create \
  --remote-ip 0.0.0.0/0 \
  --protocol icmp \
  --ingress $sec_group
openstack security group rule create \
  --remote-ip 0.0.0.0/0 \
  --dst-port 22 \
  --protocol tcp \
  --ingress $sec_group
openstack router create external_router
openstack router set \
  --external-gateway external_network external_router
```

```
openstack router set \
  --external-gateway external_network router_data_analysis_a
# Configuração do cluster Hadoop
openstack floating ip list
floating_ip_pool=XXX # selecionar o ip pool
openstack flavor create m2.dn \
    --id 100 \
    --ram 4096 \
    --disk 60 \
    --vcpus 1
openstack flavor create m2.small \
    --id 101 \
    --ram 4096 \
    --disk 20 \
    --vcpus 1
openstack flavor create m2.medium \
    --id 102 \
    --ram 8192 \
    --disk 40 \
    --vcpus 2
openstack flavor create m2.large \
    --id 103 \
    --ram 16384 \
    --disk 40 \
    --vcpus 2
openstack dataprocessing node group template create \
--name cdh-m \
    --plugin cdh \
    --plugin-version 5.11.0 \
    --processes CLOUDERA_MANAGER HDFS_NAMENODE \
     HDFS_SECONDARYNAMENODE YARN_JOBHISTORY \
     YARN_RESOURCEMANAGER ZOOKEEPER_SERVER YARN_NODEMANAGER OOZIE_SERVER \
    --flavor m2.large \
    --auto-security-group \
    --autoconfig \
    --floating-ip-pool $floating_ip_pool
openstack dataprocessing node group template create \
    --name cdh-w \
    --plugin cdh \
    --plugin-version 5.11.0 \
```

```
    --processes HDFS_DATANODE YARN_NODEMANAGER \
    --flavor m2.dn \
    --auto-security-group \
    --autoconfig \
    --floating-ip-pool $floating_ip_pool
openstack dataprocessing cluster template create \
    --name cdh-3-w \
    --node-groups cdh-m:1 cdh-w:3
openstack image create ubuntu-cdh \
    --disk-format qcow2 \
    --container-format bare \
    --file $discoimagens/sahara/ubuntu_sahara_cloudera_5.11.0.qcow2
openstack dataprocessing image register ubuntu-cdh \
    --username ubuntu
openstack dataprocessing image tags add ubuntu-cdh \
    --tags cdh 5.11.0


# Comando para criar um cluster Hadoop com Cloudera
openstack dataprocessing cluster create \
    --name cdh-u \
    --cluster-template cdh-3-w \
    --user-keypair chave-admin \
    --neutron-network net_data_analysis_a \
    --image ubuntu-cdh
```

Anexo II

Rotinas de ETL

```
# Importação de Dados
sqoop \
 list-tables \
 --connect $url \
 --username $user \
 --password $pwd \
 > tables.txt
echo $(tr '\n' ',' < tables.txt ) > tables.txt
time sqoop import-all-tables \
    --connect $url \
    --username $user$ \
    --password $pwd \
    --verbose \
    --direct \
    --num-mappers 1 \
    --compression-codec snappy \
    --exclude-tables $(cat tables.txt) \
    --as-avrodatafile \
    --warehouse-dir $dir
# Data Lake
openstack container list
openstack container create dwjuris
diretorio=dwjuris-avro
hdfs dfs \
    -Dfs.swift.service.sahara.username=admin \
    -Dfs.swift.service.sahara.password=$pwd \
    -du -h swift://dwjuris.sahara/
hdfs dfs \
```

```
    -Dfs.swift.service.sahara.username=admin \
    -Dfs.swift.service.sahara.password=$pwd \
    -mkdir -p swift://dwjuris.sahara/$diretorio
time hadoop distcp \
    -Dfs.swift.service.sahara.username=admin \
    -Dfs.swift.service.sahara.password=$pwd \
    -overwrite \
    /user/hdfs/$diretorio \
    swift://dwjuris.sahara/$diretorio/
time hdfs dfs \
    -Dfs.swift.service.sahara.username=admin \
    -Dfs.swift.service.sahara.password=$pwd \
    -cp -f \
    /user/hdfs/$diretorio \
    swift://dwjuris.sahara/$diretorio
time hdfs dfs \
    -Dfs.swift.service.sahara.username=admin \
    -Dfs.swift.service.sahara.password=$pwd \
    -cp -f \
    swift://dwjuris.sahara/$diretorio \
    /user/hadoop/$diretorio
swift list dwjuris -l -p dwjuris-text/
# Copia arquivos Avro
for arquivo in $(hdfs dfs -du -h \
    $diretorio \
    |awk '{print $5}'|awk -F "/" '{print $5}')
    do
    hdfs dfs \
        -Dfs.swift.service.sahara.username=admin \
        -Dfs.swift.service.sahara.password=$pwd \
        -mkdir -p swift://dwjuris.sahara/dwjuris-avro/$arquivo

    time hdfs dfs \
        -Dfs.swift.service.sahara.username=admin \
        -Dfs.swift.service.sahara.password=$pwd \
        -cp -f \
        $dir$arquivo \
        swift://dwjuris.sahara/dwjuris-avro/
done
```

Anexo III

Scripts para Análise de Dados

```
import com.databricks.spark.avro._
import com.databricks.spark.csv._

val movimentacao_av="/user/hadoop/dwjuris-avro/TB_FATO_MOVIMENTACAO"
val df_mov_av = spark.read.avro(movimentacao_av)
val movimentacao_av_sw="swift://dwjuris.sahara/dwjuris-avro/TB_FATO_MOVIMENTACAO/"
val df_mov_av_sw = spark.read.avro(movimentacao_av_sw)

df_mov_av_sw.limit(1000000).write.avro("swift://dwjuris.sahara/dwjuris-avro/TB_MOV_1M")
df_mov_av.groupBy($"codigomovimento").count.show
df_mov_av_sw.groupBy($"codigomovimento").count.show

val processo_av="/user/hadoop/dwjuris-avro/TB_DIM_PROCESSO"
val processo_av_sw="swift://dwjuris.sahara/dwjuris-avro/TB_DIM_PROCESSO/"
val df_processo_av_sw = spark.read.avro(processo_av_sw)
val df_mov_av = spark.read.avro(movimentacao_av)

df_mov_av.groupBy($"classeprocessual").count.show
```