
UNIVERSIDADE FEDERAL DO MARANHÃO

Curso de Ciência da Computação

Lucas Gabriel Rezende de Jesus

# Mineração de Dados em Rede Social Baseada em uma Arquitetura em Nuvem

São Luís

2018

Lucas Gabriel Rezende de Jesus

# Mineração de Dados em Rede Social Baseada em uma Arquitetura em Nuvem

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Mário Antonio Meireles Teixeira

São Luís

2018

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Rezende de Jesus, Lucas Gabriel.
Mineração de dados em rede social baseada em uma arquitetura nuvem / Lucas Gabriel Rezende de Jesus. - 2018.
53 f.

Orientador(a): Mário Antonio Meireles Teixeira.
Monografia (Graduação) - Curso de Ciência da Computação, Universidade Federal do Maranhão, Auditório DEINF - UFMA, 2018.

1. Computação em Nuvem. 2. Mineração de Dados. 3. Redes Sociais. 4. Twitter. I. Teixeira, Mário Antonio Meireles. II. Título.

Lucas Gabriel Rezende de Jesus

# Mineração de Dados em Rede Social Baseada em uma Arquitetura em Nuvem

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado em: São Luís, 11 de Julho de 2018.

**Prof. Dr. Mário Antonio Meireles Teixeira**
Orientador
Universidade Federal do Maranhão

**Profa. Dra. Simara Vieira da Rocha**
Examinador 1
Universidade Federal do Maranhão

**Prof. Dr. Samyr Béliche Vale**
Examinador 2
Universidade Federal do Maranhão

São Luís
2018

Aos meus pais,
que sobre tudo e todos
me fizeram chegar onde estou.

# Agradecimentos

Sobre tudo quero agradecer a Deus que com todo seu amor sempre me protegeu do perigo, me acolheu nos momentos de tristeza, de dor e sorriu comigo nos dias de alegria. Obrigado por estar escrevendo comigo a minha história, botando as pessoas certas no meu caminho e me proporcionando experiências enriquecedoras.

Agradeço aos meus pais, Marcos e Rejane, que vão muito além de seus papéis de pai e mãe, são verdadeiros conselheiros, guardiões e protetores da vida, sempre botando a minha educação e saúde em primeiro lugar. Me ensinaram desde pequeno a ser vitorioso, a correr atrás dos meus objetivos e dar o meu melhor sempre. Hoje sou resultado de seus esforços e serei eternamente grato por esses dois anjos na minha vida.

Aos meus familiares, minha irmã Sara, minhas avós Celeste, Graça e Luiza, minhas tias Regiana, Rossana e Renata, meus primos João e Maria Clara, meus tios Josiel e Ademílson, minha madrinha Fátima, meu padrinho Areolino, meu padastro Raimundo, minha madastra Jaqueline e a todos que de sangue ou não estão unidos nesse núcleo familiar.

Agradeço a todo corpo docente do curso de Ciência da Computação da Universidade Federal do Maranhão pela formação excelente e em especial meu orientador Prof. Dr. Mário Meireles, que com total maestria me forneceu todo apoio e suporte ao desenvolvimento desse trabalho.

Aos meus amigos da vida que o curso proporcionou, são estes Júlia, Alexsandro, André, Márcio, Ricardo, Alysson, Marcos, Tarcio, Rodrigo, Erick, Lucas, Alexandre, Feliphe, Dillsy e todos que fizeram valer a pena as noites de sonos perdidas, os sorrisos espontâneos e todo companheirismo de uma união que mais se assemelha à uma família. Obrigado programadores. Aos companheiros do Programa de Educação Tutorial (PET) e Diretório Acadêmico de Computação que me proporcionaram experiências ímpares durante minha graduação. Aos colegas do laboratório Laws (*Laboratory Of Advanced Web Systems*) pelas dicas e auxílios.

Aos meus velhos amigos de escola, Hugo, Alessandra, Bernardo, Catarina, Renan, Paulo, Cláudio, Gabriel, Damasceno, Rodrigo, Fernanda, Lígia, Mariana, Alline, Raquel, Airi e todos que trilhando ou não por caminhos diferentes, ainda permanecem unidos por um forte sentimento de irmandade.

Agradeço a Ana, Heloísa, Vinícius, Ivana e a todos que apesar da distância mandam energias positivas e torcem para o meu sucesso.

Agradeço imensamente aos meus companheiros de trabalho, Fábio, Luiz, Bruno, Flávio, Anderson, Hugo e Gustavo que foram bem além do campo profissional, se tornando verdadeiros amigos, companheiros e pessoas com que eu me orgulho de vivenciar. Obrigado por sempre me ajudarem em momentos de dificuldades.

Agradeço em especial aos meus chefes Francisco, Jhonatan e Dr. Cícero, pela oportunidade em trabalhar na equipe, pelo apoio e credibilidade aos meus estudos, pela flexibilidade e por todo crescimento profissional proporcionado pelo convívio na secretaria. Aos meus companheiros da SUINF (Supervisão de Informática da Secretaria de Segurança Pública do Maranhão), Fredereico, Wellingthon, Daniel, Kenia, Jamerson e todos que me incentivam.

Obrigado a todos que de alguma forma me ajudaram nesta conquista.

*"A imaginação é mais*
*importante que a ciência, porque a*
*ciência é limitada, ao passo que a*
*imaginação abrange o mundo inteiro."*

Albert Einstein

# Resumo

Enormes quantidades de dados são gerados diariamente ao redor do mundo. Provenientes das mais diversas origens, essas informações quase sempre estão dispostas de forma heterogênea e dispersas, sem um real valor agregado à elas. Nesse contexto, as redes sociais vem chamado atenção por serem um ambiente em que cada vez mais usuários estão dispostos a expressar suas opiniões e ideias sobre determinando assunto. Dentre elas o *Twitter* se destaca por sua simplicidade, os textos são rápidos e objetivos por terem um número de caracteres limitados. Assim, tornou-se benéfico extrair esses dados e processá-los a fim de gerar conhecimento aplicável tanto a área científica quanto comercial. Associado a isso, a chamada Computação em nuvem tem sido apontada como solução para problemas de desempenho e escalabilidade em sistemas que necessitam processar grandes quantidades de dados. Diante desse cenário, o objetivo deste trabalho é propor uma metodologia para mineração de dados de redes socais baseada em uma arquitetura em nuvem, de forma a contribuir com a literatura já existente e a fornecer uma alternativa à modelos e arquiteturas tradicionais de processamento de dados. Serão apresentados os conceitos relacionados ao processo de descoberta de conhecimentos e as técnicas empregadas em sua implementação. Por fim, é proposto um estudo de caso e feita uma avaliação da eficácia do modelo.

**Palavras-chave**: Mineração de Dados; Computação em Nuvem; Redes Sociais; Twitter.

# Abstract

Huge amounts of data are generated daily around the world. Coming from diverse origins, this information is almost always arranged in a heterogeneous and dispersed way, without a real added value to them. In this context, social networks deserve attention because they are an environment in which more and more users are willing to express their opinions and ideas about certain subjects. Among them *Twitter* stands out for its simplicity, the texts are fast and objective because they have a limited number of characters. Thus, it has been beneficial to extract this data and process it in order to generate knowledge applicable in both scientific and commercial areas. Associated with this, so-called cloud computing has been pointed out as the solution to performance and scalability problems in systems that need to process large amounts of data. In this scenario, the objective of this work is to propose a methodology for Data Mining of social networks based on a cloud architecture, in order to contribute with the existing literature and to provide an alternative to traditional models and architectures of data processing. The concepts related to knowledge discovery process and the techniques used in its implementation are presented. Finally, a case study is proposed and and evaluation of the efficacy of the model is performed.

**Keywords**: Data Mining; Cloud computing; Social Networks; Twitter.

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| API | Interface de programação de aplicações |
| AWS | Amazon Web Services |
| BI | Business Intelligence. |
| EC2 | Amazon Elastic Compute Cloud. |
| EMR | Amazon Elastic MapReduce. |
| Iaas | Infraestrutura como serviço. |
| KDD | Processo de descoberta de conhecimento |
| NIST | Instituto Nacional de Padrões e Tecnologia do Departamento de Comércio Norte-Americano. |
| NLTK | Natural language toolkit |
| Paas | Plataforma como serviço. |
| RDD | Conjunto de dados distribuído resiliente. |
| S3 | Serviço Simples de Armazenamento. |
| Saas | Software como serviço. |

# Sumário

# 1 INTRODUÇÃO

Segundo estudos de 2016 da *Business Software Alliance* (BSA) aproximadamente 2,5 quintilhões de byte são gerados diariamente no mundo. Essa enorme quantidade de dados caracterizadas principalmente pelo o seu grande volume e diversidade é denominada *"Big Data"* e podem ser oriundas das mais variadas fontes, tais como sensores, publicações em sites, imagens digitais, vídeos, sinal de GPS e das chamadas redes sociais (SRI; ANUSHA, 2016).

Marteleto (2001) define redes sociais como sendo um conjunto de participantes autônomos, que unem ideias e recursos em torno de valores compartilhados. No âmbito da informática, esse conceito pode ser entendido como ambientes virtuais que possibilitam a criação e compartilhamento de conteúdo, por meio da divulgação de ideias, interesses e opiniões. Com o aumento da população mundial e a explosão da internet como meio de comunicação humana, por conta da criação de novas tecnologias e políticas que facilitaram o seu acesso, a presença das redes sociais na vida das pessoas aumentou significantemente nestes últimos anos (CORRÊA et al., 2017).

Dentre os mais diversos serviços disponíveis o *Twitter* chama atenção por sua simplicidade e objetividade, o usuário pode enviar e receber mensagens com até 280 caracteres denominados *tweets*. Seu recente crescimento aponta a necessidade de pessoas utilizarem a internet para expressar sua opinião de forma rápida sobre diversos assuntos. A exemplo disso, em 2018 o *Twitter* foi amplamente usado na vinculação de notícias sobre a Copa do Mundo (HENRIQUES, 2018) e em 2015 e 2016, nas manifestações sobre o impeachment da ex-presidente Dilma Roussef (PENTEADO; GUERBALI, 2018).

Dessa forma, esses espaços virtuais estão sendo cada vez mais importantes fontes de informações, de modo que estudos e investimentos estão continuamente sendo direcionados na tentativa de encontrar valores nesse montante de dados (BOTHOS; APOSTOLOU; MENTZAS, 2010). Armbrust et al. (2009) ainda acrescenta que "Uma fatia crescente dos recursos de computação é gasta na compreensão de clientes, cadeias de suprimentos, hábitos de compra, classificação e assim por diante". Portanto, a mudança de mentalidade provocada pelo crescente interesse de empresas e organizações em realizar análise sobre seus dados, tem exigido soluções eficientes tanto no que diz respeito ao software quanto a arquitetura.

Nesse contexto, se tornou desafio coletar, armazenar e finalmente processar esses massivos dados. Diante desse cenário, duas novas abordagens computacionais são apontadas como possível solução para o processamento desses dados de forma eficiente, a Mineração de dados e a Computação em nuvem.

Denominada por alguns autores como KDD (*Knowledge Discovery in Database*) (FAYYAD et al., 1996), a mineração de dados permite extrair conhecimento de informações armazenadas por meio de um conjunto de etapas. Associado às redes sociais, este ciclo permite deduzir valores dos textos publicados pelos usuários, de formar a auxiliar a tomada de decisão e compreensão das opiniões de uma determinada população acerca de um produto, empresa ou assunto.

Já a chamada *Cloud Computing*, ou Computação em nuvem, é um modelo que permite o compartilhamento de recursos computacionais por demanda, de modo a facilitar o acesso, a alocação e o gerenciamento das entidades Vaquero et al. (2008). Pretendendo principalmente a escalabilidade e a abstração de recursos, a Computação em Nuvem vem revolucionando a forma que os sistemas computacionais estão sendo pensados.

Uma arquitetura proposta por Marz e Warren (2015) visa atender tanto o processamento de dados em lote quanto em velocidade. Segundo o autor, a arquitetura dividida em camadas, visa além de suprir as necessidades de um sistema robusto que possa compreender uma ampla carga de trabalho, responder com rapidez a fluxo de dados com baixa latência.

Portanto, tendo em vista a mineração de dados eficiente no intuito de geração de conhecimento, neste trabalho propõe-se um modelo arquitetural escalável e tolerante a falhas baseado em ambiente de nuvem para viabilizar e favorecer o uso de sistemas complexos de processamento de dados, diminuindo o gargalo gerado pelo uso de arquiteturas e sistemas de armazenamento e processamento de dados comuns.

## 1.1 Trabalhos Relacionados

Capua, Nardo e Petrosino (2015) propõe uma implementação da arquitetura Lambda modificada para realizar técnicas de aprendizado de máquina. A fim de executar a análise de sentimento em grandes fluxos de dados fornecidos pelo *Twitter*, o autor compara métodos de *Deep Learning* e NLP (*Natural Language Processing*). Ainda são apresentados algumas ferramentas utilizadas durante na metodologia como o *Apache Hadoop* e o *Apache park*. Kiran et al. (2015) desenvolve uma proposta de implantação da arquitetura Lambda utilizando serviços de provedores em nuvem.

O trabalho de Capua, Nardo e Petrosino (2015) apresenta um modelo para predição de bolsa de valores baseado em mineração de dados que também faz uso de NLP porém somado à métodos de aprendizagem supervisionada, como máquinas de vetor de suporte. Não obstante, o caso de estudo presente em Ozdemir (2016) encoraja o uso de técnicas de ciência de dados para tentar produzir um modelo capaz de auxiliar na previsão de preços das ações baseada em mídias sociais.

Corrêa et al. (2017) usa diferentes abordagens para a análise de sentimentos expressos por usuários do *Twitter* em relação aos filmes indicados à categoria de Melhor Filme do Oscar 2017 após avaliar as diversas abordagens, foi escolhido o *Naive Bayes Multinomial* como algoritmo de classificação. Galante e Oliveira (2008) realizam um estudo sobre mineração de dados em redes sociais, dando enfoque no processo de descoberta de conhecimento. O autor realiza um comparativo de técnicas, métodos e algoritmos adotados por artigos que serviram de base para seu trabalho.

## 1.2 Objetivos

O objetivo principal deste trabalho é propor a implantação de uma arquitetura de mineração de dados textuais de redes socais baseado em um ambiente em nuvem. Para tanto, serão empregadas ferramentas para distribuição e análise de dados bem como bibliotecas de processamento textual, ambas aplicadas mediante serviços de provedores na nuvem. Ainda destaca-se como objetivos específicos deste trabalho:

- Propor uma metologia para processamento de dados em ambiente nuvem;
- Estudar sistemas de processamento de dados;
- Realizar mineração de dados em textos provindos de redes socais;
- Aplicar ferramentas de processamento textuais;
- Utilizar serviços de provedores nuvem;
- Integrar o processo de descoberta de conhecimento na arquitetura lambda.

## 1.3 Estrutura do Trabalho

Este trabalho está organizado em 7 capítulos com suas respectivas seções e subseções. A divisão procura esclarecer todas etapas do processo de desenvolvimento bem como as observações e conceitos envolvidos nele. A princípio, no capítulo 2 é apresentado uma síntese sobre o conceito de *Big Data* e posteriormente é explanado o processamento de extração de conhecimento e sua relação com a análise de sentimento, fazendo um paralelo com a utilização destes no trabalho.

O capítulo 3 apresenta a abordagem arquitetural utilizada na metodologia. São demonstradas suas camadas e como elas podem auxiliar no problema de mineração de dados. No capítulo 4 são discutidos os conceitos relacionados à Computação em Nuvem e são detalhados os serviços utilizados na implementação.

O capítulo 5 aborda as ferramentas de processamento de dados utilizadas na metodologia. No capítulo 6 é apresentado a proposta para a mineração de dados e feita toda elucidação das etapas do processo de extração de conhecimento. Ainda é demonstrado sua implementação em ambiente nuvem mediante um estudo de caso. Ao final, o capítulo 7 contempla a conclusão que reporta a importância deste trabalho, os resultados obtidos nele e as sugestões para trabalhos futuros.

# 2 BIG DATA

Segundo NIST (2017), *Big Data* refere-se a à incapacidade de arquiteturas de dados tradicionais para lidar eficientemente com os novos conjuntos de dados com características especificas comumente conhecidas por 4V's:

- Volume: Refere-se ao tamanho do conjunto de dados
- Variedade: Dados heterogêneos de diferentes fontes, tais como dados de sensores, de dispositivos móveis, dados de rede sociais, dados geográficos, áudios, imagens entre outros.
- Velocidade: Relacionado com a taxa de fluxo.
- Valor: Refere-se às características e valores extraídos do conjunto dos dados

Diante disso, é possível notar que a definição remete à necessidade de uma arquitetura de sistema que possa ser dimensionada para atingir o desempenho e a eficiência de custos necessários, sendo essa uma das principais problemáticas relacionadas à utilização de *Big Data*.

Fundamentalmente, existem dois métodos de escalonamento de sistemas. O escalonamento vertical que consiste em aumentar os parâmetros de velocidade, processamento, armazenamento e memória do sistema. E o escalonamento horizontal, que visa utilizar recursos individuais distribuídos integrados para atuar como um único sistema, normalmente referido como sistema distribuído. Nota-se que o primeiro método é limitado por capacidades físicas no que diz respeito ao hardware, sendo o segundo tido como alternativa para os sistemas de *Big Data*.

Dessa forma, o paradigma para cargas de trabalho de *Big Data*, consiste na distribuição de sistemas de dados horizontalmente por meio de recursos independentes e acoplados para alcançar a escalabilidade necessária para o processamento de conjuntos de dados extensivos (NIST, 2017).

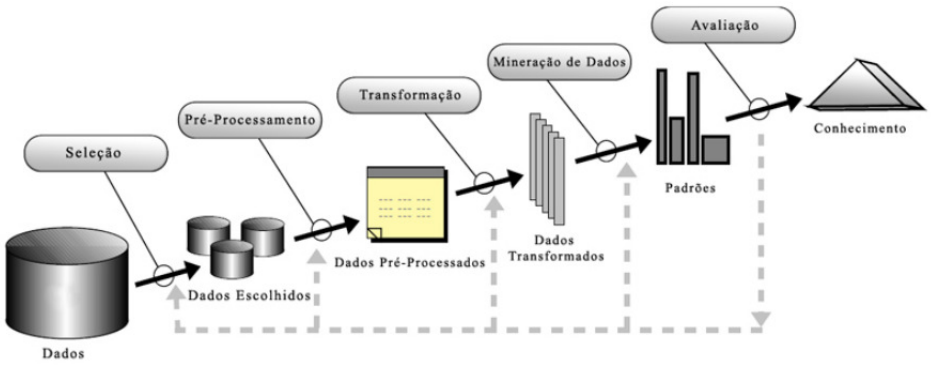
## 2.1 Processo de Descoberta de Conhecimento

A extração do conhecimento é uma área dinâmica, envolvendo integrações com outras áreas do conhecimento como Estatística, Inteligência Artificial e Banco de Dados (CAMILO; SILVA, 2009). O nome se refere ao ciclo em que os dados são submetidos à processos transformando em conhecimento (NIST, 2017). Portanto, se caracteriza por ser um conjunto de processos que busca gerar conhecimento confiável, compreensível e

potencialmente útil. Os padrões extraídos devem permitir o uso do conhecimento de forma a tirar proveito de alguma vantagem, seja científica ou comercial, por meio da otimização de processos e reduções de custos (DANTAS et al., 2008).

De acordo com Fayyad et al. (1996), o processo de descoberta de conhecimento é divido em algumas fases. Por mais que seja uma sequência de passos, o processo como um todo é tido como interativo e iterativo. Interativo pois o usuário pode controlar o curso das atividades realizando intervenções, e iterativo porque é uma sucessão finita de operações onde o resultado de cada uma é dependente dos resultados das que a precedem (PRASS, 2016). A Figura 1 mostra uma visão geral de cada uma das fases.

Figura 1 – Processo de descoberta de conhecimento




Fonte: (CAMILO; SILVA, 2009).

- Seleção: O KDD começa na fase de seleção, onde itens específicos dentro de um domínio que contém todas as possíveis variáveis (características ou atributos) são selecionados para o resto da análise. Esse processo pode ser bastante complexo, pois os dados podem ser oriundos de diversas fontes diferentes (API, *Data Warehouse*, Planilhas) (PRASS, 2016).

- Pré-processamento: O pré-processamento corrige as inconsistências encontradas nos dados para garantir a confiabilidade dos resultados da mineração (COSTA et al., 2016). Nesta etapa deverão ser realizadas tarefas que eliminem dados redundantes, recuperem dados incompletos e avaliem possíveis dados discrepantes ao conjunto (PRASS, 2016).

- Transformação: Fase responsável pelo armazenamento dos dados de forma que eles possam ser utilizados eficientemente na etapa de mineração.

- Mineração de Dados (*Data Mining*): Principal etapa do processo de descoberta de conhecimento, responsável pela produção de padrões por meio da exploração e análise de dados. Tamanha é sua importância que ainda não há um consenso

sobre as nomenclaturas, sendo possível encontrar na literatura o termo *Data Mining* para descrever todo o ciclo (COSTA et al., 2016). Diante desse fato, a seção 2.2 é destinada exclusivamente para este processo.

- Avaliação: Etapa final em que os dados são disponibilizados já processados. O conhecimento adquirido através da técnica de *Data Mining* deve ser interpretado e avaliado para que seja gerado um valor e o objetivo final seja alcançado. Caso o resultado não seja satisfatório, o processo pode retornar a qualquer um dos estágios anteriores ou até mesmo ser recomeçado (PRASS, 2016).

## 2.2 Data Mining

Berry e Linoff (1997) definem *Data Mining*, como a exploração e análise, de forma automática ou semi-automática, de grandes bases de dados com objetivo de descobrir padrões e regras. Dessa forma, a função da mineração de dados no processo de descoberta do conhecimento é encontrar padrões e valores que estão escondidos no enorme conjunto de dados e interpretá-los para conhecimento e informações úteis (RAVAL, 2012). Nesta Fase, a garimpagem dos dados deve acontecer após todo o pré- processamento das grandes quantidades de dados que, tratadas e sumarizadas, estarão livres de ruídos e conterão somente os dados relevantes à pesquisa a ser feita (COSTA et al., 2016).

Hegland (2001) classifica as técnicas de *Data Mining* em algumas classes com diferentes objetivos:

- *Induction*: encontrar padrões e equações que caracterizam os dados.
- *Compression*: diminuir a complexidade dos dados, substituindo por conceitos simples.
- *Querying*: encontrar melhores maneiras de consultar os dados.
- *Approximation*: encontrar modelos para aproximar as observações.
- *Search*: buscar padrões recorrentes.

De acordo com Fayyad et al. (1996) existem diversos métodos de *Data Mining* para encontrar respostas ou extrair conhecimento em repositórios de dados. Dentre essas técnicas, duas tiveram papéis importantes na construção da metodologia proposta: associação e classificação.

### 2.2.1 Associação

Geralmente ocorre no processo de encontrar relacionamentos entre diferentes atributos em grandes bancos de dados de clientes. A ideia é encontrar a natureza das causalidades

entre os valores dos diferentes atributos. Também podem ser aplicadas na ligação entre ocorrências por um único evento. Como no exemplo dado em Aggarwal e Philip (1999), que considera as compras feitas por clientes em um supermercado e que a partir disso pode-se traçar uma relação entre os preços dos produtos e o comportamento de compra dos clientes.

### 2.2.2 Classificação

Classifica um item a uma ou várias classes categóricas pré-definidas, objetivando envolver a descrição gráfica ou algébrica das características diferenciais das observações de várias populações, além da classificação das observações, em uma ou mais classes pré-determinadas. Comumente é um método clássico baseado em técnicas de Aprendizado de Máquina (RAVAL, 2012).

## 2.3 Análise de Sentimento e *Machine Learning*

Liu (2012) define Análise de Sentimento, ou mineração de opiniões, como a área de estudo que analisa as opiniões, sentimentos, avaliações e atitudes das pessoas com relação a diferentes entidades, que podem ser produtos, serviços, organizações, individuais, eventos, tópicos, problemas e seus respectivos atributos. Nesse cenário, a análise de sentimento é representada pela classificação do texto de forma positiva, negativa ou neutra e envolve o uso de técnicas de Aprendizado de Máquina (*Machine Learning*) (CORRÊA et al., 2017). Nesse cenário, a análise de sentimento é representada pela classificação do texto expresso nos *tweets*, de forma positiva, negativa ou neutra e envolve o uso de técnicas de Aprendizado de Máquina.

Uma dessas técnicas é o Aprendizado Supervisionado, que consiste no uso de um conjunto de treinamento criado para treinar um classificador (BAEZA-YATES; RIBEIRO-NETO, 2013). Segundo os autores, um algoritmo de classificação é considerado supervisionado quando um conjunto de treinamento é usado para treinar o classificador. Para os algoritmos não supervisionados, não é necessário que sejam fornecidos exemplos de treinamento.

### 2.3.1 Naive Bayes

O algoritmo *Naive Bayes* é um dos métodos de aprendizado supervisionado mais utilizados no escopo de análise de sentimentos devido ao seu desempenho notável na classificação de textos e análise de sentimentos (BAEZA-YATES; RIBEIRO-NETO, 2013).

O classificador é considerado ingênuo por assumir que os atributos são incondicionalmente independentes, de forma que um evento não influencia em outro, ou

seja, indecência forte entre os recursos. Apesar dessa característica simplista, o classificador reporta um dos melhores desempenhos em tarefas de classificação textual (MILIDIÚ, 2006).

O *Naive Bayes* é fundamentado no Teorema de Bayes 2.1, definida como: dados dois conjuntos, A e B, que determina a probabilidade da hipótese A estar em B, dado um conjunto de evidências B (CORRÊA et al., 2017).

$$P(A|B) = \frac{P(B|A)P(A)}{P(A)} \tag{2.1}$$

Os classificadores Bayesianos podem ser abordados de duas principais formas, modelos binário e multinomial (CORRÊA et al., 2017). No modelo binário, cada documento é representado por um vetor de atributos binário de modo que cada atributo indica a ocorrência ou não, respectivamente 0 ou 1. Já no modelo multinomial, cada documento é representado por um vetor de atributos inteiros, indicando o número de vezes que cada termo (*feature*) acorre no documento (BAEZA-YATES; RIBEIRO-NETO, 2013).

### 2.3.2 Ferramentas de Análise Textual

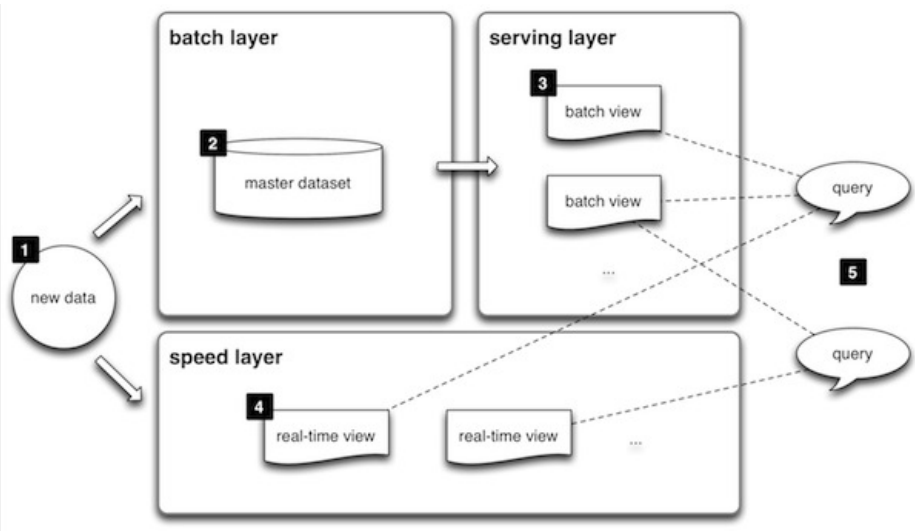
Na implantação da arquitetura proposta neste trabalho, foi escolhido o *TextBlob* como ferramente de análise textual. O *TextBlob* é uma biblioteca Python (2 e 3) para processamento de dados textuais. Ela possui uma API simples para mergulhar em tarefas comuns de processamento de linguagem natural e é integrada com a plataforma NLTK (*Natural Language Toolkit*) (TEXTBLOB, 2018). O módulo de análise de sentimento é treinado pelo algoritmo *Naive Bayes* usando características padrões do analisador, extraídas de um *DataSet* interno em inglês.

Ele utiliza o modelo multinomial do classificador, dessa forma, após todo processo de análise, o módulo retorna um valor dentro do intervalo [-1.0, 1.0] representando a polaridade da sentença, com valores mais próximos de -1 como sentimento negativo e próximos de 1 como positivo.

# 3 ARQUITETURA LAMBDA

A arquitetura lambda formulada em Marz e Warren (2015) é uma proposta que visa atender tanto o processamento de dados em lote quanto em tempo real. Segundo os autores, a arquitetura dividida em camadas visa suprir as necessidades de um sistema que seja tolerante a falhas e que possa compreender uma ampla carga de trabalho aproveitando os métodos de processamento de dados em lote e fluxo. A abordagem mescla a capacidade de tolerância a falhas e alta precisão sobre o conjunto de dados permitidas pelo processamento em lote e a baixa latência e alta taxa de transferência decorrentes do processamento de fluxo. Dessa forma ambas camadas são complementares. A Figura 2 apresenta uma visão geral da arquitetura Lambda e sua divisão em 3 principais camadas.

Figura 2 – Visão geral da arquitetura Lambda




Fonte: (HAUSENBLAS; BIJNENS, 2017).

## 3.1 Camada Batch

Esta etapa prevê o armazenamento de dados de entrada em um conjunto de dados principal, que permanece imutável e realiza a computação dos dados continuamente, o que atribui uma precisão perfeita, pois é capaz de processar todos os dados disponíveis recorrigindo qualquer erro e atualizando as exibições existentes. A camada de lote é executada em um loop e continuamente recompila o lote gerando novas visões *batch*. De forma geral, é mais simples expressar cálculos robustos na camada em lote.

## 3.2 Camada Speed

Esta camada é responsável pelo processamento dos dados em velocidade, de forma a compensar a alta latência da camada em lote, fornecendo visões com alta taxa de atualização e em tempo tão rápido quanto necessário para os requisitos de uma aplicação. Esta etapa produz visualizações conforme os dados são recebidos, portanto a grande diferença entre as camadas é o corpo de dados analisados. Enquanto a camada em lote recalcula todos os dados de única só vez, a camada de velocidade computa somente os dados mais recentes, de forma incremental, o que permite alcançar os resultados com menor latência possível.

## 3.3 Camada Serving

As saídas da camada em lote e velocidade são indexadas na camada de serviço. Os dados então são mesclados para que possam ser consultados por diferentes visões. Habitualmente esta camada é um banco de dados escalável que suporta atualizações em lote e leituras aleatórias. Por conta da latência, da camada *batch*, alguns resultados da camada serving também possuem o mesmo atraso.

# 4 COMPUTAÇÃO EM NUVEM

Neste capítulo são discutidos os principais conceitos relacionados à Computação em Nuvem, partindo de definições e características passando pelos modelos de serviços, modelos de implantação e por fim apresentar o provedor de nuvem adotado, a *Amazon Web Services* (AWS).

## 4.1 Definição

A Computação em Nuvem é apontada como um novo paradigma para o fornecimento de infraestrutura computacional. Vaquero et al. (2008) a define como um grande conjunto de recursos virtualizados facilmente utilizáveis e acessíveis tais como *hardware*, plataformas de desenvolvimento e serviços. Tais recursos podem ser dinamicamente reconfigurados de modo a se ajustar a uma carga variável de trabalho, permitindo sua otimização.

Outra importante definição e amplamente adotada é encontrada em Mell, Grance et al. (2011) que definem Computação em Nuvem como um modelo que permite acesso onipresente, conveniente e sob demanda a uma rede compartilhada e a um conjunto de recursos de computação configuráveis podendo ser provisionado e liberado rapidamente com esforço mínimo de gerenciamento ou interação com o provedor de serviços.

Dessa forma, é compreendido que esse novo paradigma abstrai grande parte da preocupação com administração da infraestrutura por meio do provimento de recursos sob demanda e custos proporcionais a utilização.

## 4.2 Características

Existe uma série de pontos importantes sobre essa nova tecnologia, dentre os quais podem ser citados cinco características essenciais (MELL; GRANCE et al., 2011):

- Serviços sobre demanda: É a capacidade do usuário provisionar recursos computacionais, como tempo do servidor, instâncias de maquinas virtuais e armazenamento em rede, conforme necessário, sem uma interação direta com o provedor de serviço.
- Amplo acesso à rede: Os recursos estão disponíveis na rede e acessados por meio de mecanismos que permitem o uso por diversas plataformas, de forma que seja necessário somente o acesso à *Internet* para a sua utilização.

- Agrupamento de Recursos: É a possibilidade dos serviços serem acessíveis simultaneamente por vários consumidores com múltiplos recursos alocados dinamicamente conforme a demanda. Isso permite que o usuário especifique os recursos em nível mais alto de abstração sem ter conhecimento sobre a localização e funcionamento exato dos mesmos.

- Elasticidade Rápida: Os recursos podem ser alocados e liberados elasticamente, em alguns casos automaticamente, de forma escalável. Dessa forma, para os consumidores, os recursos apresentam-se de forma ilimitada, de modo que os recursos podem ser provisionados a qualquer momento e com indefinida capacidade.

- Serviço Mensurável: Os sistemas em nuvem controlam e otimizam automaticamente o uso de recursos por meio de ferramentas de medição em algum nível de serviço, como por exemplo, armazenamento, processamento, largura de banda e contas de usuários ativas. Assim sendo, o uso dos recursos pode ser monitorado, controlado e relatado, proporcionando transparência para o provedor e o consumidor.

## 4.3 Modelo de Serviço

De acordo com a definição do NIST (Instituto Nacional de Padrões e Tecnologia do Departamento de Comércio Norte-Americano), são três os Modelos de Serviços que compõem a Computação em Nuvem. Essa definição é amplamente aceita na literatura e é comumente referida como uma pilha de camadas: infraestrutura, plataforma e software como serviço, onde não necessariamente estão relacionadas entre si (ALCATTAN, 2014).

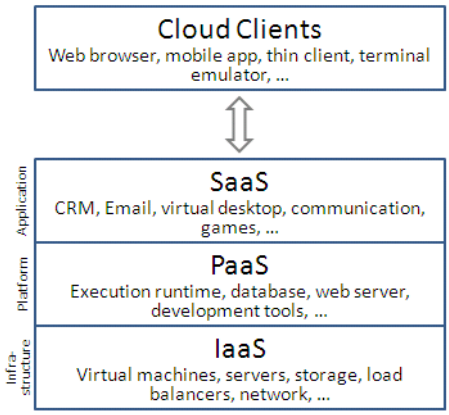
- Software como Serviço (SaaS): É a capacidade do provedor de oferecer ao consumidor aplicativos como serviço. Estes são acessíveis a partir de dispositivos por meio de interfaces como um navegador da web ou módulos de programas. O usuário não gerencia ou controla diretamente a infraestrutura por trás dos recursos como rede, servidores, sistemas operacionais, armazenamento ou capacidade de aplicação individual.

- Plataforma como Serviço (PaaS): Nesse cenário o usuário pode instalar e gerenciar suas próprias aplicações. Isso significa que toda infraestrutura e tecnologia (sistemas operacionais, ambientes de desenvolvimento) são fornecidos no intuito do usuário, normalmente um programador, possa configurar suas aplicações e os aspectos referentes ao ambiente utilizado por elas. Exemplos desse tipo de serviço incluem o *Google App Engine* e *Windows Azure*.

- Infraestrutura como Serviço (IaaS): Nesta camada é oferecido ao usuário um conjunto de recursos computacionais de baixo nível tais como armazenamento,

processamento e redes pelos quais se tem a liberdade de instalar, executar e configurar aplicações, sistemas operacionais ou qualquer outro tipo de *software* suportado. Por se tratar de um ambiente virtualizado, essa infraestrutura é escalável permitindo ser dinamicamente alocada de acordo com as necessidades da aplicação. Esse modelo disponibiliza os recursos necessários para as camadas anteriores (Paas e Saas), como visto na Figura 3. Portanto o usuário tem um maior controle sobre o armazenamento, sistemas e aplicativos implantados sem ter conhecimento explícito sobre o funcionamento interno da infraestrutura.

Figura 3 – Modelos de serviço de Computação em Nuvem organizados em pilha.




Fonte: (ALCATTAN, 2014).

## 4.4 Modelo de Implantação

Em relação à disponibilidade, existem diferentes tipos de modelos de implantação para os ambientes de Computação em Nuvem. A restrição ou abertura de acesso depende do processo de negócios, do tipo de informação e do nível de visão desejado (COUTINHO et al., 2013), conforme descrito a seguir :

- Nuvem privada: É uma infraestrutura em nuvem utilizada exclusivamente por uma organização, sendo esta local ou remota e administrada pela própria empresa ou por terceiros. Neste modelo os recursos não são compartilhados dessa forma utiliza-se servidores dedicados.
- Nuvem pública: Sua infraestrutura é disponibilizada para o público em geral, sendo acessada por qualquer usuário que conheça a localização do serviço. Usualmente são serviços disponibilizados pela *Internet*.
- Nuvem comunitária: Fornece uma infraestrutura compartilhada por várias organizações com interesses e objetivos em comum.

- Nuvem híbrida: É a composição de dois ou mais modelos de nuvens, que podem ser do tipo privada, pública ou comunitária de modo que sejam entidades únicas, porém conectadas por meio de tecnologia proprietária ou padronizada que permite a portabilidade de dados e aplicações.

## 4.5 Provedores de Nuvem

O provedor de nuvem é responsável por fornecer recursos virtualizados, de acordo com os modelos descritos anteriormente. Cada provedor define como um determinado ambiente será disponibilizado ao consumidor, ou seja, quais são os padrões, aplicações e tecnologias que estarão disponíveis (MARTINS et al., 2016). Neste trabalho é utilizado o *Amazon Web Services* (AWS) que será detalhado na seção seguinte.

### 4.5.1 Amazon Web Services

O *Amazon Web Services* é uma coleção de serviços e recursos de computação remota que juntos formam uma plataforma de computação em nuvem, oferecida pela internet. Dentro dos diversos serviço,s se destacam a hospedagem de aplicação, backup e armazenamento, distribuição de conteúdo, big data e análises (AMAZON, 2015).

Dentre os benefícios fornecidos pela plataforma da AWS se destaca a agilidade, flexibilidade e a elasticidade instantânea. Este último refere-se a capacidade de poder implantar novos aplicativos e expandir instantaneamente à medida que sua carga de trabalho cresce e instantaneamente reduzir com base na demanda. Outro fator preponderante em sua escolha foi o uso do modelo denominado *"Pay as You Go"*, uma abordagem de definição de preços proporcional ao uso dos serviços.

A infraestrutura da Nuvem AWS é baseada em regiões que correspondem às localizações geográficas e zonas de disponibilidade (AZs). As regiões oferecem várias zonas de disponibilidade separadas e isoladas fisicamente, que são conectadas com baixa latência, alta taxa de transferência e redes altamente redundantes. Essas zonas fornecem aos usuários uma maneira mais fácil e eficiente de projetar e operar aplicativos e bancos de dados, tornando-os mais altamente disponíveis, tolerantes a falhas e com ajuste de escala em comparação com infraestruturas tradicionais como os "data centers" (AWS. . . , 2018).

#### 4.5.1.1 Serviços AWS

A *Amazon Web Services* disponibiliza um grande conjunto de produtos classificados em categorias como computação, armazenamento, bancos de dados, análise, redes, dispositivos móveis, ferramentas de desenvolvedor, ferramentas de gerenciamento, IoT (*Internet of Things*), segurança e aplicações empresariais. Os seguintes serviços foram utilizados no desenvolvimento deste trabalho:

- *Amazon Elastic Compute Cloud (Amazon EC2)* : É um ambiente de computação virtual que permite através de interface *web* a criação de máquinas virtuais com diversas configurações de *hardware* e Sistemas Operacionais. Possui elasticidade por meio da alocação de recursos dinamicamente ou não, o que permite um ambiente adaptativo às aplicações. As instâncias são classificadas de acordo com a combinação do tipo de armazenamento, quantidade de memória, CPU e recursos de rede.

- *Amazon S3 (Simple Storage Service)*: É um repositório de objetos de alta disponibilidade e durabilidade. Foi criado para armazenar e recuperar qualquer quantidade de dados de qualquer local. Os dados são armazenados em pastas dentro de *buckets*, organizados hierarquicamente. Atualmente é o serviço de armazenamento na nuvem que conta com o maior suporte do mercado e da comunidade em geral.

- *Amazon EMR (Amazon Elastic MapReduce)*: Disponibiliza uma estrutura gerenciada pelo *Hadoop* que processa grandes quantidades de dados em um *cluster* escalável de instâncias de servidores virtuais da *Amazon Elastic Compute Cloud (EC2)*.O EMR também permite executar outras estruturas distribuídas conhecidas, como *Apache Spark*, *HBase*, *Presto*, e *Flink* bem como a interação com plataformas de armazenamento de dados da AWS, como *Amazon S3*, *Amazon DynamoDB* e *Amazon Redshift*.

- *Amazon Redshift*: É a solução de *data warehouse*, ou seja um repositório central de informações, da AWS. Ela permite a análise dos dados por meio de consultas SQL padrão ou ferramentas de *Business Intelligence*. Possui integração facilitada com os *buckets S3* e é habilitado para *Redshift Spectrum* que ajusta automaticamente a capacidade computacional de consultas com base nos dados que estão sendo recuperados. Através dela é possível criar um *cluster* de banco de dados escalável e de alto desempenho.

# 5 PROCESSAMENTO DE DADOS

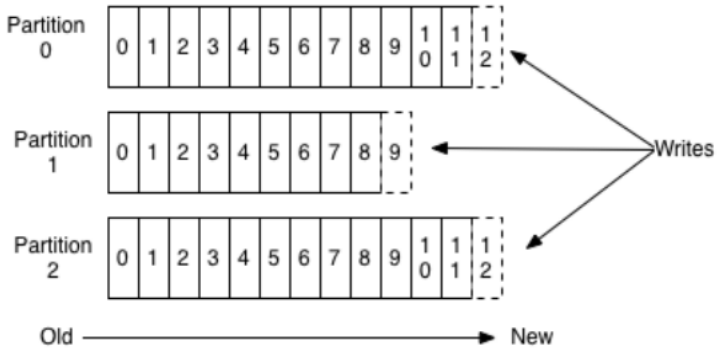
Neste capítulo serão discutidas as ferramentas e técnicas de processamento e de dados utilizadas na metodologia deste trabalho. Num primeiro momento será apresentada a plataforma de distribuição de mensagens e logo após o *Apache Hadoop* em comparativo com o *Apache Spark*, sendo explanado o porquê da escolha deste último como o sistema central de processamento de dados da arquitetura proposta bem como sua estrutura, ecossistema. Por fim, é demonstrado o mecanismo de tratamento de dados em fluxo e seu modelo de programação.

## 5.1 Kafka

*Apache Kafka* é uma plataforma distribuída de mensagens e *streaming*. Habitualmente é utilizada em processamento de fluxo, acompanhamento de atividades no site, coleta e monitoramento de métricas entre outras atividades (APACHE. . . , 2017). O *Kafka* possui alguns componentes que possibilitam o seu funcionamento de forma escalonável e tolerante a falhas (HORTONWORKS, 2017), são eles:

- Mensagem: É a unidade básica do *Kafka*. Todo fluxo de dados se resume a mensagens que podem ser escritas em *string* ou algum outro formato de texto como o JSON.
- Tópico: O tópico é uma forma de categorizar os tipos de mensagens. Para cada tópico, o *cluster Kafka* mantém um *log* de confirmação estruturado com uma ou mais partições. Um tópico pode possuir varias partições, porém ao receber uma nova mensagem o *Kafka* automaticamente direciona aquela mensagem para uma partição específica dependendo de sua chave (*key*) mantendo a leitura ordenada das mensagens.
- Produtor: É o componente responsável por produzir e enviar as mensagens para um tópico específico. Ao receber uma mensagem em um tópico, o *Kafka* direciona a mensagem para uma partição, garantindo sempre a ordem de produção das mesmas, como observável na Figura 4 disposta abaixo.
- Consumidor: Os consumidores são processos que aguardam um ou mais tópicos a publicação de mensagens. Essa classe também é responsável por acompanhar quais mensagens já foram armazenas ou deslocados das partições. O próprio *Kafka* mantém todas as mensagens em disco por um período de tempo configurável.
- *Broker*: É o componente central do funcionamento do *Kafka*. Consiste em uma instância *Kafka* em uma máquina. Um conjunto de vários *Brokers* formam um *Kafka*

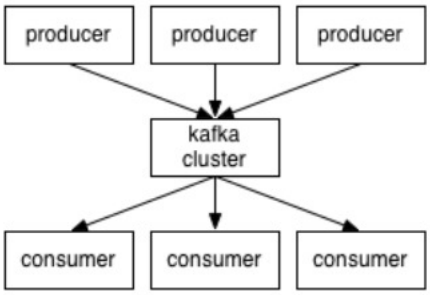
Figura 4 – Funcionamento das partições de um tópico Kafka.




Fonte: (HORTONWORKS, 2017).

*Cluster*, o que permite escalabilidade da solução. Os produtores enviam mensagens para o *cluster* que por sua vez aloca e repassa os dados aos consumidores. Cada *Broker* gerencia a persistência e a replicação de dados de mensagens. A Figura 5, demonstrada a seguir, apresenta de forma geral o funcionamento de um *Kafka Cluster*.

Figura 5 – Funcionamento de um *Kafka Cluster*.




Fonte:(HORTONWORKS, 2017).

## 5.2 Hadoop

O *Apache Hadoop* é uma biblioteca de *software* de código aberto projetada para realizar armazenamento e processamento distribuído de dados em um *cluster* de computadores. Este *framework* acopla um sistema de arquivos distribuído chamado *Hadoop Distributed File System* (HDFS) e o *MapReduce* para o processamento de dados (HADOOP, 2016).

### 5.2.1 Hadoop Distributed File System

O HDFS é um sistema de arquivos distribuídos projetado para ser executado em *hardware* comum. Possui muitas semelhanças com sistemas de arquivos distribuídos existentes. No entanto, as diferenças de outros sistemas de arquivos distribuídos são significativas. HDFS é tolerante a falhas, permite acesso de dados *streaming*, além de suportar grandes conjuntos de dados sem comprometer as taxas de transferências (BORTHAKUR et al., 2008).

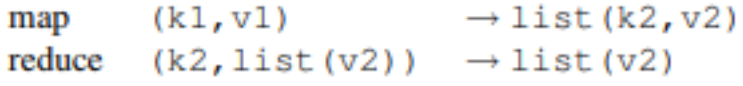
### 5.2.2 MapReduce

*MapReduce* é um modelo de programação e uma implementação associada para processar e gerar grandes conjuntos de dados. A computação é feita por meio do cálculo que leva um conjunto de pares de chave/valor de entrada e produz um conjunto de pares de chave/valor de saída. O usuário mediante uma biblioteca *MapReduce* expressa o cálculo como duas funções: Mapear e Reduzir (DEAN; GHEMAWAT, 2008).

A Mapear pega um par de entrada e produz um conjunto de pares chave/valor intermediários. A biblioteca *MapReduce* agrupa todos os valores intermediários associados com a mesma chave intermediária I e passa-os para a função Reduzir.

A função Reduzir aceita a chave intermediária e um conjunto de valores para essa chave, os valores são mesclados para formar um grupo possivelmente menor de valores. Normalmente, as saídas são apenas apenas zero ou um. Os valores intermediários são fornecidos para a função de redução do usuário por meio de um iterador. A Figura 6 apresenta uma visão lógica das funções *Map* e *Reduce*.

Figura 6 – Visão Lógica das funções Mapear e Reduzir.




Fonte: (DEAN; GHEMAWAT, 2008).

Dessa forma, os programas desenvolvidos com o *Hadoop* gozam do processamento e armazenamento paralelo. Uma tarefa é divida em várias pequenas tarefas que são executadas em paralelo por diferentes máquinas dentro de um *cluster* e então agregadas para alcançar o resultado final.

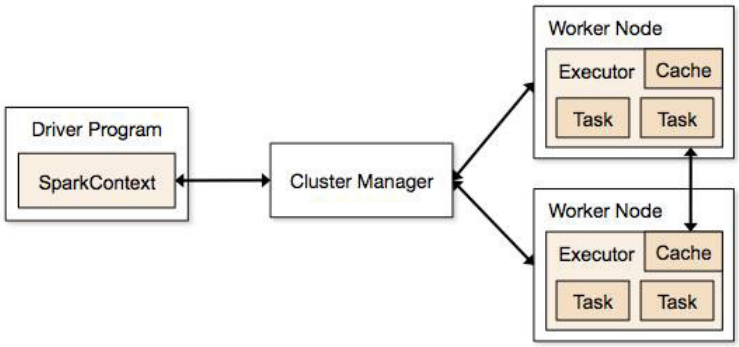
## 5.3 Spark

O *Spark* é um sistema de processamento de dados distribuído e altamente escalável baseado em *Java Virtual Machine* (JVM). Seu grande diferencial fica por conta da sua

velocidade, pela qual a *Apache* afirma ser capaz de "executar programas até 100x mais rápido que o *Hadoop MapReduce* na memória, ou 10x mais rápido no disco". Isso se deve ao fato da maior parte do processamento ser realizado na memória principal dos nós do *cluster*. Enquanto o *Hadoop* armazena os resultados intermediários do processamento, em disco, o *Spark* armazena em memória. Dessa forma, o *Spark* estende as funcionalidades do *MapReduce*, suportando tarefas mais eficientes de computação como consultas iterativas e processamento de *data streams*. O mecanismo fornece uma estrutura abrangente e unificada capaz de gerenciar requisitos de processamento de grandes conjuntos de dados de diversas naturezas, como textos, imagens, conteúdo de vídeo, dados gráficos e das mais variadas fontes (dados de aplicativos *on-line*, em lote e em tempo real) (THOTTUVAIKKATUMANA, 2016).

### 5.3.1 Arquitetura do Spark

Figura 7 – Arquitetura *Spark*




Fonte: (SPARK. . . , 2018).

A arquitetura das aplicações *Spark*, apresentada na Figura 7, são executadas como processos independentes em um *cluster*. O *SparkContext* é o responsável pela conexão da aplicação ao *Spark* e com o gerenciador do *cluster*, responsável pela alocação de recurso. Feito isso, o *Spark* envia o código da aplicação para os nós executores que realizam os cálculos e armazenam os dados do programa. Após isso, o *SparkContext* envia as tarefas para os executores que permanecem ativos durante todo ciclo de vida do aplicativo. Isso permitir isolar aplicações uma das outras, pois podem se instanciados vários contextos para executores diferentes. De forma geral, a arquitetura de uma aplicação *Spark* é constituída por:

- *Driver Program*: é o processo principal que gerencia a criação dos nós e coordena o encaminhamento de tarefas para serem executados.
- *Cluster Manager*: componente responsável pela administração das máquinas utilizados como *Workers Node*. É utilizada quando o *Spark* executa em *cluster*.

- *Worker Node*: são as máquinas que executarão as tarefas coordenadas pelo *Driver Program*, sendo responsável pelo processamento e gerenciamento de dados da aplicação implementada.

### 5.3.2 Modelo de Programação

O principal componente do modelo de programação do *Spark* é o RDD (*Resilient Distributed Dataset*) (ZAHARIA et al., 2012). Este e suas operações somado ao já citado *Spark Context* representam o modelo de programação do *Spark*. Os RDDs são motivados por dois tipos de aplicações que as atuais estruturas de computação são ineficientes: algoritmos iterativos e dados interativos e ferramentas de mineração. O *Spark* armazena dados em RDDs em diferentes partições. Estes por serem mantidos em memória otimizam o processamento de dados e a reorganização dos cálculos. Eles também são tolerantes a falhas pois trabalham com transformações em memória compartilhadas podendo recriar e recalcular os conjuntos de dados. Outra importante característica é sua imutabilidade, um RDD pode ser modificado com operações de transformações porém a estrutura original permanece a mesma, sendo retornado um novo conjunto de dados (PENCHIKALA, 2015). Dois tipos de operações são suportadas por esse componente:

- Transformações: retorna um novo RDD baseado na transformação utilizada. Algumas das transformações são: *map, filter, flatMap, groupByKey, reduceByKey, aggregateByKey.*

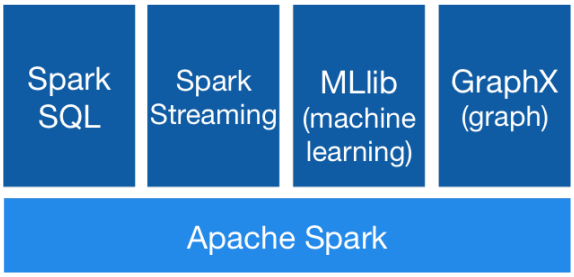
- Ações: as operações de ação retornam um valor. Quando uma função de ação é chamado em um objeto RDD, todas as consultas de processamento de dados são calculadas e o valor resultante é retornado. Alguns exemplos de ações são: *reduce, collect, count, first, take, countByKey e foreach.*

### 5.3.3 Ecosistema *Spark*

O *frameWork* disponibiliza uma gama de componentes que se acoplam ao mecanismo principal, *Spark Core*, responsável pelas funções básicas de processamento, como *map*, *reduce*, entre outros. As bibliotecas fornecem várias funcionalidades úteis para diversos cenários de processamento de *big data*, uma visão geral é vista na Figura 8 (PENCHIKALA, 2015).

- *Spark Streaming*: utilizado para processar dados de fluxo em tempo real. É baseado no estilo de micro-processamento de computação e processamento. Ele usa um tipo de abstração chamada *DStream*, que consiste em uma série de RDDs, para processar os dados em tempo real.

Figura 8 – Ecosistema *Spark*




Fonte: (SPARK. . . , 2018).

- *Spark SQL*: permite consultas semelhantes a SQL sobre os conjuntos de dados do *Spark* por meio de consultas tradicionais e ferramentas de visualização. O *Spark SQL* permite que os usuários extraiam seus dados em formatos diferentes (como JSON, ou um banco de dados), além de executar transformações e ações.
- *Spark* MLlib: é a biblioteca de aprendizado de máquina. Possui diversos algoritmos e ferramentas de aprendizagem comuns, tais como classificação, regressão, agrupamento, filtragem colaborativa, redução de dimensionalidade.
- *Spark GraphX*: é a API do *Spark* que realiza processamento sobre grafos. Para suportar computação de grafos, o GraphX expõe um conjunto de operadores fundamentais ( *subgraph*, *joinVertices* e *aggregateMessages*) além de incluir algoritmos e construtores para simplificar análise gráfica.

#### 5.3.3.1 Spark Structured Streaming

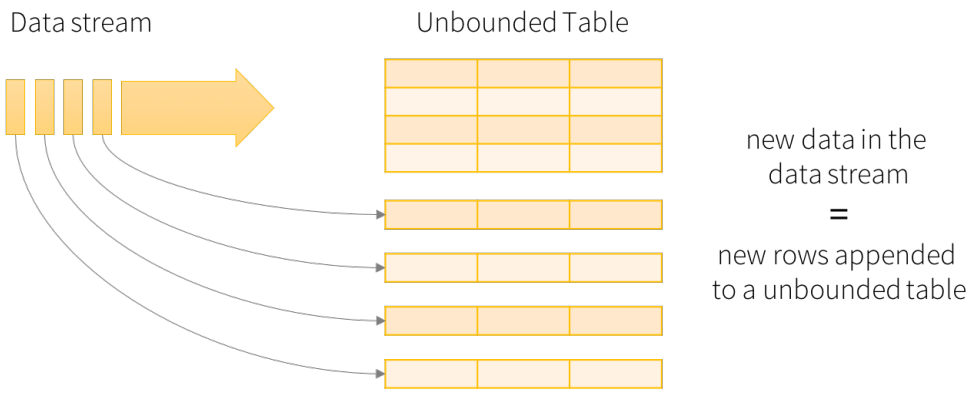
O Structured Streaming é um mecanismo de processamento de fluxo dimensionável e tolerante a falhas criado a partir do componente *Spark SQL*. O grande diferencial desse método é a possibilidade de expressar a computação de streaming da mesma forma da em lote com dados estáticos. O mecanismo *Spark SQL* é responsável por executar a computação de forma incremental, de modo que os resultados sejam atualizados à medida que o fluxo de dados contínuo seja recebido (SOMPOLSKI, 2017).

##### 5.3.3.1.1 Modelo de Programação

O fluxo de dados de entrada é tratado como tabela de entrada que está sendo anexada continuamente alimentada. Cada item de dados que chega ao fluxo é como uma nova linha sendo anexada à tabela, como demosntrado na Figura 9.

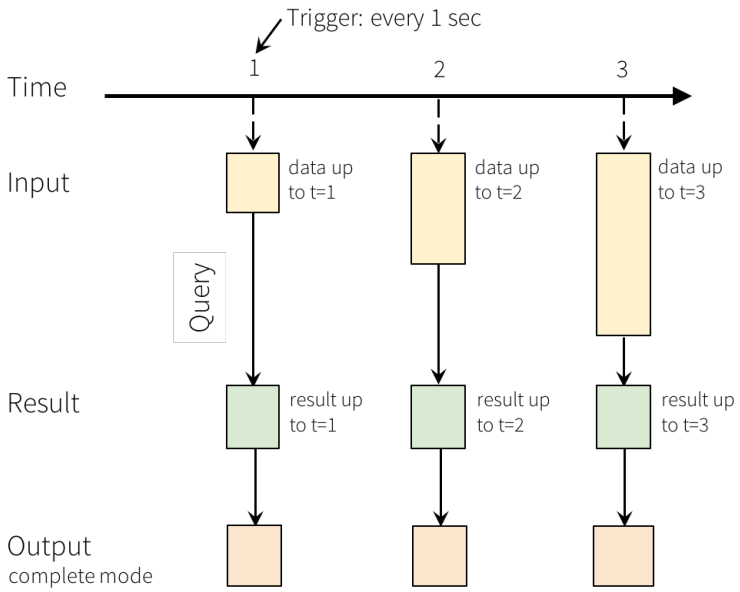
Uma consulta na tabela de entrada gera uma tabela de resultados. Em intervalo definido de um gatilho, novas linhas são adicionadas à tabela de entrada, que consequentemente atualiza a tabela de resultados. Esta estrutura deve ser gravada em um armazenamento externo. O *Spark* possibilita 3 modos diferentes de saída:

Figura 9 – Fluxo de dados como tabela




Fonte: (STRUCTURED. . . , 2018).

Figura 10 – Modelo de Programação do Structured Streaming




Fonte: (STRUCTURED. . . , 2018).

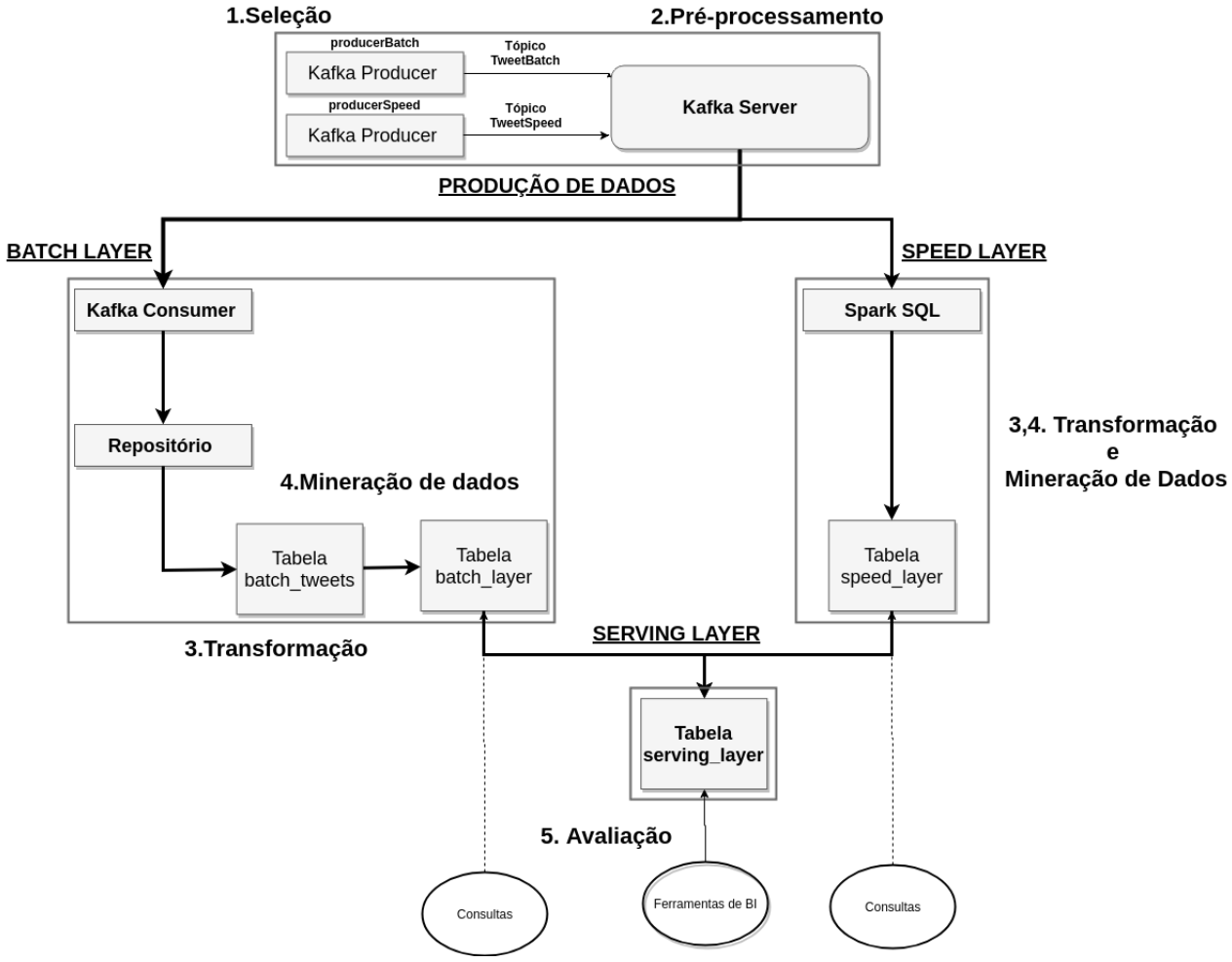
- *Complete Mode*: toda tabela de resultados atualizada será gravada no armazenamento externo. A Figura 10 é um exemplo do modo.

- *Append Mode*: Somente as novas linhas anexadas na Tabela de Resultados, desde o último gatilho, serão gravadas no armazenamento externo.
- *Update Mode*: Somente as linhas que foram atualizadas na Tabela de resultados desde o último gatilho serão gravadas no armazenamento externo, ou seja, irá exibir apenas as linhas que foram alteradas desde o último gatilho.

# 6 PROPOSTA DE ARQUITETURA PARA MINERAÇÃO DE DADOS

Neste capítulo são apresentados os métodos desenvolvidos para a implementação da arquitetura lambda afim de realizar o processo de descoberta de conhecimento em rede social. Serão descritos todos os procedimentos presentes nas etapa da metodologia deste trabalho bem como a explanação de cada camada da arquitetura. O sistema foi construído utilizando soluções computacionais em ambiente nuvem e a API da rede social *Twitter* como fonte de dados. Cabe destacar as escolhas do Pyhton e Scala como linguagens de programação, ambas são amplamente empregadas em projetos da área de ciência de dados devido a compatibilidade com *frameworks* e facilidades na manipulação dos dados. A Figura 11 apresenta um diagrama geral da arquitetura proposta.

Figura 11 – Esquema da Proposta




Fonte: O autor

## 6.1 Produção dos Dados

Esta camada atua como a fonte de dados do resto da arquitetura. Na metodologia presente neste trabalho, ela é responsável por fazer a seleção das variáveis de interesse da rede social, pelo pré-processamento destes dados e transmissão. Nesta etapa é utilizado o *Kafka* que permite a criação de fluxos distribuídos para as diferentes camadas da arquitetura. Diante desse cenário foi criado um fluxo de registros para a camada *batch* e outro para a camada *speed*, agrupados em dois tópicos diferentes.

Uma vantagem da implementação é a possibilidade de formular mensagens diferentes provenientes da mesma fonte de dados. Diante desse fato, é optado pelo envio do campo texto somente para a camada *batch* por seu propósito de permitir uma análise mais completa sobre o conjunto de dados. Em contrapartida, na camada de velocidade a baixa latência e alta taxa de atualização dos dados impediria a leitura do texto, sendo mais objetivo o envio dos outros dados da mensagem.

### 6.1.1 Seleção

Todos os novos dados são provenientes da rede social *Twitter*. Um dos fatores de sua escolha foi a robustez e completude de sua API, que mesmo no nível gratuito (*Standard*) oferece diversas operações e funções aos usuários (TWITTER, 2018). Nessa etapa foi utilizado a ferramenta Tweepy (TWEEPY, 2018), que forneceu acesso aos métodos da API do *Twitter*. A autenticação foi feita mediante a configuração de chaves de acesso e *token's* disponibilizados após o cadastro na plataforma de desenvolvedores da rede social. Todo o processo de requisição e resposta foi envelopado pela classe *tweepy.api* que abstrai solicitações *GET* e retorna uma lista de objetos correspondentes aos *tweets* em formato JSON o que facilita a manipulação dos resultados. Devido a deficiência de ferramentas para análise de sentimento em língua portuguesa, foram utilizados *tweet's* em inglês. Dentre os dados disponibilizados pelas respostas às requisições, foram escolhidos 7 valores que influenciam diretamente no processo de análise de opinião e na mineração dos dados.

- *Id*: identificador único de cada *tweet*.
- *Text*: o texto propriamente dito.
- *Created_at*: data de criação do *tweet*.
- *User.followers_count*: número de seguidores do usuário que publicou a mensagem.
- *User.location*: localização geográfica do usuário.
- *Favorite_count*: quantidade de vezes que o *tweet* recebeu marcação de favorito.
- *Retweet_count*: quantidade de vezes que o tweet foi repostado.

### 6.1.2 Pré-processamento

Todos os valores foram concatenados em uma *string* separada por ';' e codificadas com UTF-8, o que permitiu a interpretação de caracteres especiais muito presentes nos textos da rede social, característica de sua variabilidade.

#### 6.1.2.1 Análise de Sentimento

No intuito de permitir a eficiência na etapa da mineração de dados e a fim de contornar as dificuldades de se realizar o mesmo procedimento de *machine learning* em camadas diferentes, na etapa de pré-processamento é realizada a análise de sentimento do texto dos *tweets*. Isso permite que os valores de polaridade das sentenças sejam enviadas no escopo da mensagem para as camadas que realizam a mineração dos dados (*Batch Layer* e *Speed Layer*), garantindo integridade desses dados. Como citado anteriormente, foi utilizado o *TextBlob* para realizar a classificação dos textos.

Portanto, após a requisição à API do *Twitter* e feito o pré-processamento é gerado uma lista com os campos seguidos dos valores. Cada produtor *Kafka* converte a lista em *bytes* que são anexadas em seu respectivo tópico. Esse processo é feito a cada um minuto de forma assíncrona, permitindo que a mensagem fique armazenada em uma fila a espera de ser consumida.

## 6.2 Batch Layer

Nesta camada os dados de entrada são armazenados de forma permanente a fim de permitir uma precisão perfeita do processamento e a possibilidade de recálculo dos resultados. Na metodologia, todos os dados ficam contidos em repositórios de objetos na nuvem e são posteriormente transferidos para um banco de dados relacional, onde são estruturados, processados e fornecem visões completas sobre os dados provindos das etapas anteriores.

### 6.2.1 Transformação

O fluxo de mensagens é consumido por um classe consumidora do *Kafka* associado ao tópico *batch layer*. Todo esse tráfego é decodificado e armazenado em formato CSV em um *bucket* S3 da AWS organizado hierarquicamente por data. Este funciona como repositório imutável e de alta disponibilidade dos *tweets*. Logo após, os dados são enviados à uma instância do banco de dados *Redshift* e carregados em uma tabela chamada *lote_tweets*. A nomenclatura das colunas segue o mesmo padrão dos campos dos arquivos de origem.

### 6.2.2 Mineração de Dados

Neste momento são executadas consultas em formato SQL sobre os valores da tabela por meio de cálculos sobre a totalidade dos dados a fim de produzir visualizações em lote (*batch views*) e associações entre os valores. A alta precisão é traduzida na possibilidade de correção de erros por meio do recálculo sobre o conjunto completo de dados e das atualizações das exibições. São feitas operações no intervalo de uma hora sobre os *tweets* existentes e os resultados são agrupados por sua localização. Segue abaixo uma lista com tais operações:

- Média Aritmética dos valores de sentimentos.
- Contagem dos id's.
- Soma dos valores de seguidores dos usuários.
- Soma das vezes favoritadas.
- Soma das quantidade de *retweet's*.

## 6.3 *Speed Layer*

O propósito da camada de velocidade na metodologia é permitir uma análise em baixa latência dos *tweets* de forma eficiente. A ingestão dos dados é feita somente para o dia atual e sobre esses são feitas as mesmas agregações da camada em *batch*, porém em um escopo de dados diferente. Para tanto, é usado o *Spark Structured Streaming*, que possibilita o tratamento do fluxo de *tweets* de forma incremental e em intervalos de um minuto, o que proporciona resultados rápidos sobre a pesquisa de termos.

### 6.3.1 Transformação e Mineração de Dados

É iniciada uma sessão *Spark* com o *Kafka* como o formato de *stream* e feito a conexão com o tópico *speed* criado. A configuração permite com que o ponto inicial de consulta seja os registros mais recentes. Dessa forma, quando uma consulta é interrompida ela posteriormente será reiniciada a partir do ponto em que o fluxo foi interrompido, como uma espécie de fila, evitando que se carregue os mesmos registros.

Durante o processamento são feitas extrações dos valores dos *tweets* por meio de expressões SQL. Os dados são mapeados para os tipos de dados correspondentes e assim transformados em RDD's. A partir de então, a API do *Spark Structured Streaming* permite realizar diversas operações sobre esse tipo de estrutura. No cenário, são feitos as mesmas operações de agregações e agrupamento da camada em lote, ou seja, contagem, soma e média agrupados por localização. Dessa forma, o formato da saída será o mesmo, o que

possibilita a comparação dos resultados das camadas. A pesquisa por novos dados é feita a cada um minuto por meio de uma *trigger*. Os registros são exportados para a tabela *speed layer* em um banco de dados instanciado na nuvem, o que permite a visualização diária dos dados.

## 6.4 Serving Layer

Nesta etapa, a saída corresponde a mesclagem entre os resultados das camada em lote e da camada de velocidade. No cenário isso possibilita agregações de dados dos *tweets* diários e históricos, servindo como uma espécie de camada intermediária entre as consultas e os dados. Sua composição é bem simples, uma tabela de dados somente leitura com uma tabela temporária preenchida pela *batch view* e outra atualizável a cada chamada provinda da *speed view*. Como exemplo foi usada a soma dos valores dos campos de cada camada agregados pela localização, ficando da seguinte forma:

- Soma das médias dos valores de sentimento.
- Soma da contagens dos id's.
- Soma dos números de seguidores.
- Soma da contagem de favoritos.
- Soma das quantidade de retweet's

### 6.4.1 Avaliação

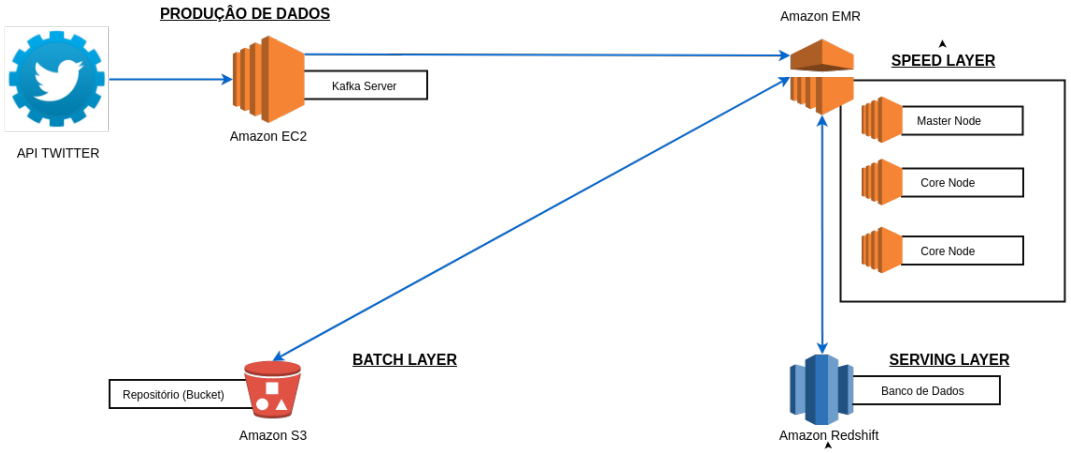
Na metodologia proposta, a etapa de avaliação é servida pelos resultados dos processos das camadas *batch*, *speed* e *serving*. Diante dos dados, podem ser utilizadas ferramentas de BI para a análise das informações permitindo o auxílio de decisões e melhorias de desempenho ou simples consultas estruturadas às tabelas de resultado das camadas.

## 6.5 Implantação em Ambiente Nuvem

A implantação em ambiente nuvem visa atender os requisitos de escalabilidade e eficiência do sistema. São utilizados quatro serviços diferentes do provedor da AWS. O diagrama da Figura 12 apresenta a disposição desses serviços na arquitetura.

Uma máquina virtual EC2 é instanciada a fim de exercer a produção dos dados. O *Kafka Server* permanece fazendo requisições a API do *Twitter*. Neste momento, um programa Pyhton realiza a seleção das variáveis e o pré-processamento.

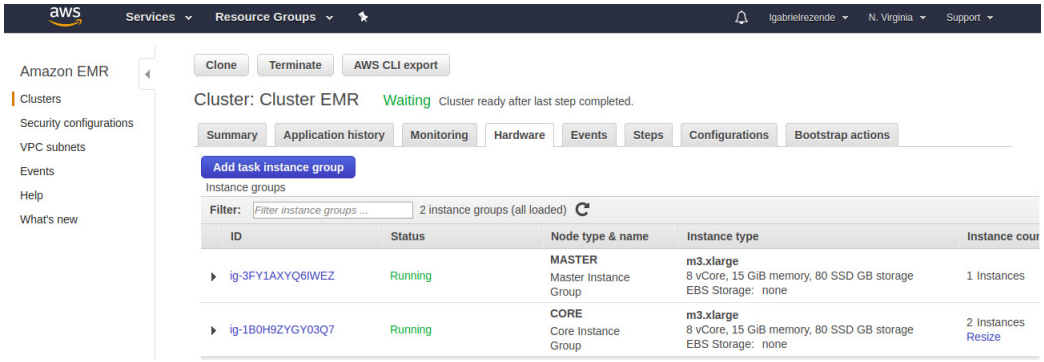
Figura 12 – Implementação com serviços da AWS




Fonte: O autor

Como visto na Figura 13, um *Cluster EMR* com três nós, sendo um master e dois core, realiza o processamento das outras três camadas. Cabe destacar que os nodes são máquinas virtuais, portanto são instâncias de EC2. Uma aplicação Pyhton é responsável pelo envio do *tweets* ao *bucket S3* (repositório) e pela comunicação com o banco de dados *RedShift*. Este recebe as consultas que são executadas e geram as visões em lote.

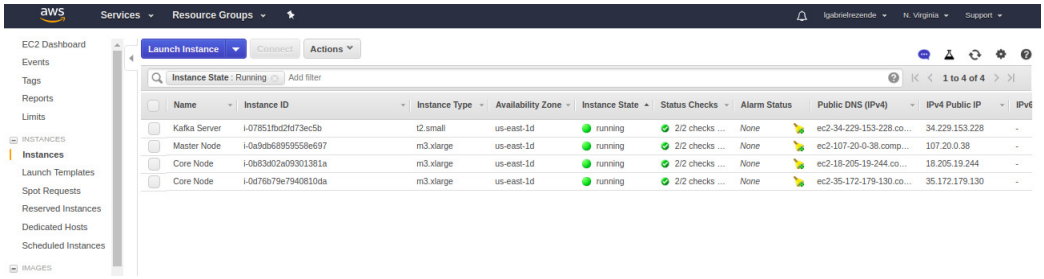
Figura 13 – Interface do cluster EMR




Fonte: O autor

O *Spark* instancia um master node e dois core node na estrutura do *cluster*. Este conjunto de máquinas virtuais (Figura 14) realiza o processamento em velocidade que em um intervalo definido repassa os resultados a uma tabela do banco de dados.

Figura 14 – Máquinas virtuais EC2




Fonte: O autor

O ponto central do *serving layer* é o banco de dados *RedShift*. Desse modo, uma aplicação pode realizar mesclagens dos dados de forma a obter uma visão diferente e mais completa sobre o escopo tratado. No cenário, um código Pyhton realiza esse procedimento como exemplo.

### 6.5.1 Estudo de Caso

Como estudo de caso, foram pesquisados por termos relacionados a grandes empresas presentes na bolsa de valores de Nova York (*New York Stock Exchange*). Devido a limitação do algoritmo da análise de sentimento, a língua escolhida é a inglesa. Na tabela 1 é apresentado o formato de colunas das tabelas resultados das camadas da arquitetura.Na tabela 2 é apresentado o formato da tabela *batch_tweets*, responsável por manter todos os valores dos *tweets* armazenados antes de serem processados na camada *batch*.

Tabela 1 – Formato de tabelas do banco de dados

| location | avg_sentiment | count_id | sum_followers_count | sum_favorite_count | sum_retweet_count |
|---|---|---|---|---|---|

Tabela 2 – Formato da tabela *batch_tweets*

| id | text | sentiment | created_at | followers_count | retweet_count | favorite_count | location |
|---|---|---|---|---|---|---|---|

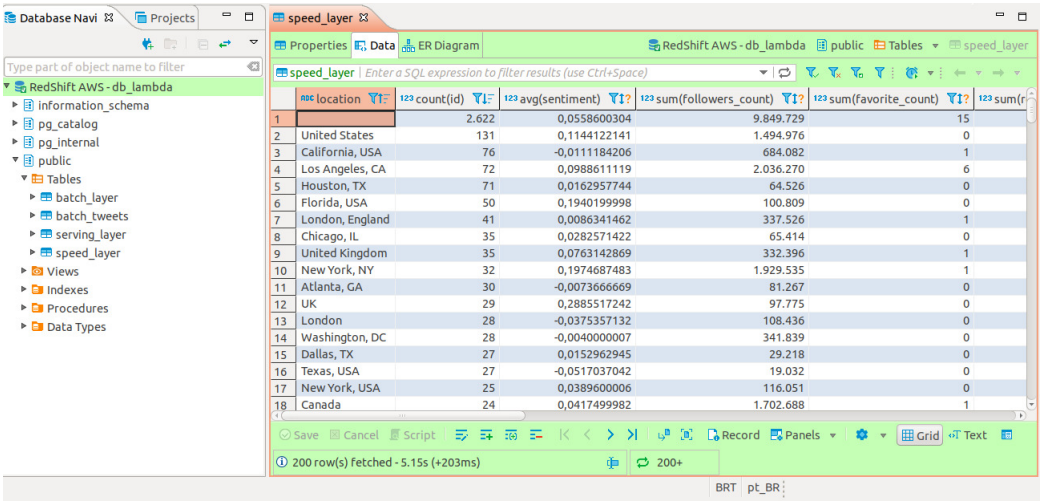
Foram rodados experimentos no ambiente da AWS com instancias de máquinas virtuais seguindo configurações de *hardware* sugeridas pelo próprio provedor. Existe uma extensa lista presente em Amazon... (2018) com diversos tipos de instâncias, preços e recursos computacionais. A tabela 3 apresenta as configurações utilizadas.

Tabela 3 – Configurações de instâncias EC2

| Tipo de instância | Configuração | Quantidade |
|---|---|---|
| m3.xlarge | 8 vCore, 15 GiB memória, 80 SSD GB | 3 |
| t2.small | 1 vCore, 2Gib memória | 1 |

A Figura 15 apresenta a disposição dos primeiros registros, ordenados por localização e quantidade, em um experimento de duração de 7,2 horas. Nesse intervalo foram processados 8847 *tweets*. No exemplo, a pesquisa foi feita pelo termo *"Apple"*.

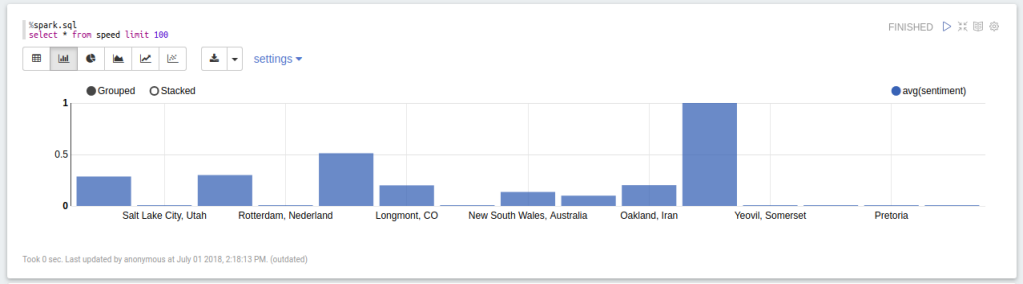
Figura 15 – Disposição dos registros na tabela *speed_layer*




Fonte: O autor

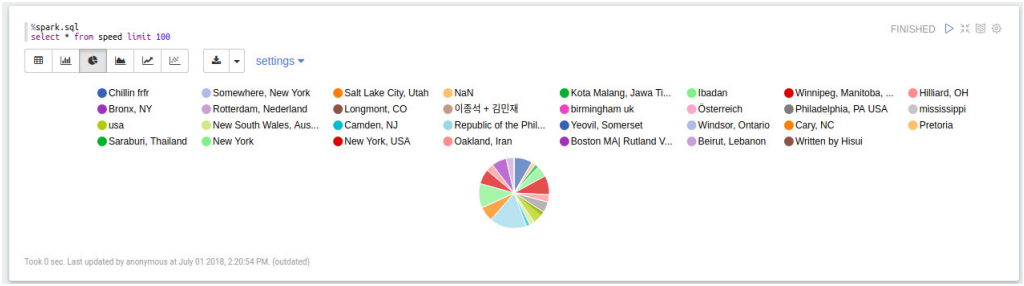
Os dados podem então ser consultados por meio de ferramentas de consultas, permitindo associações sobre os resultados e assim auxiliar em processos de tomada de decisão ou otimização. Como exemplo, foram traçados gráficos com o interpretador Zeppelin (APACHE, 2018). A Figura 16 apresenta uma gráfico de médias de valores dos sentimentos, a imagem posterior dispõe das localizações organizadas em pizza.

Figura 16 – Gráfico em barra




Fonte: O autor

Figura 17 – Gráfico em pizza




Fonte: O autor

## 6.6 Considerações Finais

Desse modo, a arquitetura em nuvem permitiu a ingestão de dados do *Twitter* de forma eficiente. A configuração de *Cluster* e máquinas adotadas suportou todo processamento durante o intervalo de experimentação. A utilização de CPU do nó master responsável pelo gerenciamento das tarefas permaneceu em 10% e dos 16 vCores (núcleos de máquinas virtuais) foram utilizados 4.

As operações em disco foram minimizadas por conta do processamento em memória principal adotado pelo modelo distribuído do *Spark*. Este cenário ainda se justifica quando é observado a quantidade de memória utilizada e disponível durante o experimento. De 22,50 Gb disponíveis foram utilizados 17,38 Gb.

Por fim, os gráficos e registros gerados ao final da metodologia permitem uma visualização completa sobre o conjunto de dados. A agregação dispõe de valores como a média de análise de sentimento e quantidade de *tweets* por localização o que possibilita dentro do estudo de caso a geração de valores para empresas pesquisadas e consequentemente fornece um auxílio no processo de tomada de decisão.

# 7 CONCLUSÃO

Este trabalho apresentou uma proposta de mineração de dados em redes socais baseado em ambiente nuvem. A metodologia desenvolvida realizou as etapas do processo de descoberta de conhecimento mediante uma abordagem da arquitetura lambda, para tal foram utilizadas ferramentas de análise e distribuição de dados por meio de recursos computacionais em nuvem. A proposta mostrou-se eficiente tanto no que diz respeito à contemplar os estágios da mineração quanto no desempenho do processamento dos dados.

A base da arquitetura lambda mesclados com os fundamentos do processo de descoberta de conhecimento possibilitaram a formulação da metodologia. A seleção e pré-processamento realizados na camada de produção de dados em conjunto com a análise de sentimentos dos textos foram essenciais para atenuar a dificuldade de se implementar o mesmo algoritmo de classificação em fluxos diferentes.

A transformação e mineração dos dados realizadas em camadas separadas e por ferramentas diferentes permitiu comparar o processamento em lote em velocidade. É constatado que as etapas intermediárias de armazenamento promovem um cálculo sobre um conjunto maior de dados enquanto o consumo imediato do fluxo proporciona visualizações rápidas sobre os resultados do processo global.

Sobre a ótica do processamento dos dados, o emprego de serviços hospedados em ambiente nuvem foram cruciais para o desempenho e sucesso do sistema. A alocação e elasticidade dos recursos otimizaram o funcionamento do *Cluster Spark* que apesar de consumir um grande volume de memória principal, mostrou ser eficiente nas tarefas de processamento de dados. O amplo acesso à rede permite que a aplicação e a plataforma da AWS, sejam acessíveis pela *internet* e por meio desta realizar o monitoramento e medição dos serviços alocados.

Por fim, tendo em vista o processo de mineração em redes sociais e a computação em ambiente nuvem, este trabalho atinge o objetivo de contribuir com a literatura dessas áreas por meio da exposição de um modelo de descoberta de conhecimento bem como as técnicas e princípios utilizados para a concretização do mesmo.

## 7.1 Trabalhos futuros

Diante da deficiência de ferramentas de análise textual em português, é sugerido como trabalho futuro o aprimoramento das técnicas de análise de sentimento de forma que possam realizar classificação textual em sentenças em português. Ainda é incentivado a realização de simulações com modelos diferentes da arquitetura lambda e serviços distintos de provedores

nuvem. A elaboração de uma aplicação de BI forneceria visualizações diferentes sobres os resultados e estenderia as funcionalidades do sistema onde possivelmente se aplicará à diversas outras áreas de estudo.

# Referências

AGGARWAL, C. C.; PHILIP, S. Y. Data mining techniques for associations, clustering and classification. In: SPRINGER. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Yorktown Heights, New York, 1999. p. 13–23. Citado na página 22.

ALCATTAN, R. F. Integration of cloud computing and web2. 0 collaboration technologies in e-learning. *arXiv preprint arXiv:1406.5020*, p. 48, 2014. Citado 2 vezes nas páginas 27 e 28.

AMAZON, E. Amazon web services. *Available in: http://aws. amazon. com/es/ec2/(November 2012)*, p. 1, 2015. Citado na página 29.

AMAZON EC2 tipos de instâncias - AWS. Amazon, 2018. Acessado em: 2018-06-29. Disponível em: <https://aws.amazon.com/pt/ec2/instance-types/>. Citado na página 45.

APACHE. *Apache Zeppelin*. Apache, 2018. Acessado em : 2018/06/29. Disponível em: <https://zeppelin.apache.org/>. Citado na página 46.

APACHE Kafka. Apache, 2017. Disponível em: <https://kafka.apache.org/>. Citado na página 31.

ARMBRUST, M. et al. *Above the clouds: A berkeley view of cloud computing*. Berkeley, California, 2009. Citado na página 15.

AWS global infrastructure - Amazon Web Services. Amazon, 2018. Acessado: 2018-06-20. Disponível em: <https://aws.amazon.com/pt/about-aws/global-infrastructure/>. Citado na página 29.

BAEZA-YATES, R.; RIBEIRO-NETO, B. *Recuperação de Informação-: Conceitos e Tecnologia das Máquinas de Busca*. Porto Alegre: Bookman Editora, 2013. Citado 2 vezes nas páginas 22 e 23.

BERRY, M. J.; LINOFF, G. *Data mining techniques: for marketing, sales, and customer support*. New York: John Wiley & Sons, Inc., 1997. Citado na página 21.

BORTHAKUR, D. et al. Hdfs architecture guide. *Hadoop Apache Project*, v. 53, p. 3–5, 2008. Citado na página 33.

BOTHOS, E.; APOSTOLOU, D.; MENTZAS, G. Using social media to predict future events with agent-based markets. *IEEE Intelligent Systems*, IEEE, n. 1, 2010. Citado na página 15.

CAMILO, C. O.; SILVA, J. C. d. Mineração de dados: Conceitos, tarefas, métodos e ferramentas. *Universidade Federal de Goiás (UFC)*, p. 1–29, 2009. Citado 2 vezes nas páginas 19 e 20.

CAPUA, M. D.; NARDO, E. D.; PETROSINO, A. An architecture for sentiment analysis in twitter. In: *Proceed. of Int. Conference on E-learning, Germany*. Germany: E-learning, 2015. Citado na página 16.

CORRÊA, I. T. et al. Análise dos sentimentos expressos na rede social twitter em relação aos filmes indicados ao oscar 2017. Universidade Federal de Uberlândia, 2017. Citado 4 vezes nas páginas 15, 17, 22 e 23.

COSTA, C. N. et al. Descoberta de conhecimento em bases de dados. *Revista Eletrônica: Faculdade Santos Dumont*, v. 2, 2016. Citado 2 vezes nas páginas 20 e 21.

COUTINHO, E. et al. Elasticidade em computaçao na nuvem: Uma abordagem sistemática. *XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos (SBRC 2013)-Minicursos*, p. 220–221, 2013. Citado na página 28.

DANTAS, E. R. G. et al. O uso da descoberta de conhecimento em base de dados para apoiar a tomada de decisões. *V Simpósio de Excelencia em Gestão e Tecnologia*, p. 1–10, 2008. Citado na página 20.

DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, ACM, v. 51, n. 1, p. 107–113, 2008. Citado na página 33.

FAYYAD, U. M. et al. Advances in knowledge discovery and data mining. the MIT Press, 1996. Citado 3 vezes nas páginas 16, 20 e 21.

GALANTE, R. de M.; OLIVEIRA, J. P. M. de. Um estudo sobre mineração de dados em redes sociais. 2008. Citado na página 17.

HADOOP, A. Welcome to apache hadoop. *Welcome to Apache Hadoop*, 2016. Citado na página 32.

HAUSENBLAS, M.; BIJNENS, N. *Lambda Architecture*. Luettu, 6, 2014: IEEE, 2017. Acessado: 2018-06-24. Disponível em: <http://lambda-architecture.net/>. Citado na página 24.

HEGLAND, M. Data mining techniques. *Acta numerica*, Cambridge University Press, v. 10, p. 313–355, 2001. Citado na página 21.

HENRIQUES, G. *Copa do Mundo bomba no Twitter com hashtags e jogadores; veja os mais mencionados*. Torcedores.com, 2018. Acessado em : 2018-06-29. Disponível em: <https://www.torcedores.com/noticias/2018/05/copa-twitter-hashtags-jogadores?enable-feature=new_layout>. Citado na página 15.

HORTONWORKS. *Hortonworks Data Platform: Apache Kafka Component Guide*. Santa Clara, Califórnia, 2017. Citado 2 vezes nas páginas 31 e 32.

KIRAN, M. et al. Lambda architecture for cost-effective batch and speed big data processing. In: IEEE. *Big Data (Big Data), 2015 IEEE International Conference on*. Hong Kong, 2015. p. 2785–2792. Citado na página 16.

LIU, B. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, Morgan & Claypool Publishers, v. 5, n. 1, p. 1–167, 2012. Citado na página 22.

MARTELETO, R. M. Análise de redes sociais: aplicação nos estudos de transferência da informação. *Ciência da informação*, SciELO Brasil, v. 30, n. 1, p. 71–81, 2001. Citado na página 15.

MARTINS, A. W. S. et al. Um ambiente virtual baseado em nuvem para simulação de redes de computadores com o ns-3. Universidade Federal do Maranhão, p. 19–22, 2016. Citado na página 29.

MARZ, N.; WARREN, J. *Big Data: Principles and best practices of scalable realtime data systems*. New York: Manning Publications Co., 2015. Citado 2 vezes nas páginas 16 e 24.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, p. 6, 2011. Citado na página 26.

MILIDIÚ, R. L. *Aprendizado de Máquina para o Problema de Sentiment Classification*. Tese (Doutorado) — PUC-Rio, 2006. Citado na página 23.

NIST, D. Big data interoperability framework. *Special publication 1500-1*, v. 1, 2017. Citado na página 19.

OZDEMIR, S. *Principles of Data Science*. Baltimore, Maryland: Packt Publishing Ltd, 2016. Citado na página 16.

PENCHIKALA, S. *Big data processing with apache spark*. Berkeley, California: Lulu.com, 2015. 6-12 p. Citado na página 35.

PENTEADO, C. L. de C.; GUERBALI, J. G. As manifestações do impeachment no twitter: uma análise sobre as manifestações de 2015. *Ponto-e-Vírgula: Revista de Ciências Sociais*, n. 19, 2018. Citado na página 15.

PRASS, F. S. Kdd–uma visão geral do processo. *Recuperado em*, v. 15, 2016. Citado 2 vezes nas páginas 20 e 21.

RAVAL, K. M. Data mining techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*, Citeseer, v. 2, n. 10, 2012. Citado 2 vezes nas páginas 21 e 22.

SOMPOLSKI, J. Structured streaming in apache spark: Easy, fault tolerant and scalable stream processing. *10 th Extremely Large Databases Conference (XLDB)*, databricks, p. 9–19, 2017. Citado na página 36.

SPARK Overview. Apache, 2018. Acessado: 2018-06-24. Disponível em: <https://spark.apache.org/docs/latest/>. Citado 2 vezes nas páginas 34 e 36.

SRI, P. A.; ANUSHA, M. Big data-survey. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, v. 4, n. 1, p. 74–80, 2016. Citado na página 15.

STRUCTURED Streaming Programming Guide. Apache, 2018. Disponível em: <https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html>. Citado na página 37.

TEXTBLOB. *TextBlob: Simplified Text Processing*. 2018. Disponível em: <http://textblob.readthedocs.io/en/dev/>. Citado na página 23.

THOTTUVAIKKATUMANA, R. *Apache Spark 2 for Beginners*. Phoenix, Northern California: Packt Publishing, 2016. 10 - 13 p. Citado na página 34.

TWEEPY. *Tweepy Documentation*. 2018. Acessado em: 2018/06/28. Disponível em: <http://tweepy.readthedocs.io/en/v3.5.0/>. Citado na página 40.

TWITTER. *Docs - Twitter Developers*. Twitter, 2018. Acessado em: 2018/06/28. Disponível em: <https://developer.twitter.com/en/docs.html>. Citado na página 40.

VAQUERO, L. M. et al. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, ACM, v. 39, n. 1, p. 51, 2008. Citado 2 vezes nas páginas 16 e 26.

ZAHARIA, M. et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: USENIX ASSOCIATION. *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. Berkeley, California, 2012. p. 2–2. Citado na página 35.