*[[1]](#footnote-1)Abstract*— In the Smart Cities’ contexts, one of the challenges is to process the large volume of data, which is in continuous expansion due to the constant increase of people and objects connected to the Internet. In this scenario, it is possible for the citizens to virtually participate of questions addressed by your respective local government, which is essential for the development of Smart Cities and known as eParticipation. This paper describes a process to collect, through tweets (Social Network Twitter posts), metrics related to e-Participation of the Brazilian State Capitals, mapping them and questioning the already classified as Smart Cities, utilizing for this, two of the main Data Processing tools: Apache Spark and Apache Storm.

Processamento de dados com Apache Spark e Storm: Uma visão sobre e-Participação nas capitais dos estados brasileiros

*Keywords*— eParticipation, Data Processing, Smart Cities.

# I. Introdução

A

tualmente, o crescimento populacional tem sido uma das fontes de estresse no que se refere à infraestrutura e recursos de uma cidade (CLARKE, 2013), fenômeno conhecido também como Tragédia dos Comuns (HARDIN, 1968), no qual há um cenário de alta demanda por determinados recursos finitos que quando explorados em larga escala, se tornam escassos. De acordo com o autor desse conceito, não existe solução técnica para esse problema.

No entanto, é possível utilizar Tecnologias de Informação e Comunicação (TICs), objetivando transformar os sistemas de uma cidade e otimizar o uso de seus recursos finitos, melhorando a eficiência da economia, possibilitando desenvolvimento politico, social, cultural e urbano, tornando-a uma Cidade Inteligente (SAÉZ-MARTÍN; ROSSARIO; CABA-PEREZ, 2014).

No contexto de Cidades Inteligentes, alguns objetos do nosso cotidiano têm a capacidade de serem conectados à Internet através de eletrônicos embarcados, sensores e software, formando a Internet das Coisas (IoT - Internet of Things). Em 2013, menos de 1% desses dispositivos estavam conectados, sendo a previsão para 2020 de 212 bilhões. Quanto a pessoas conectadas à Internet, prevê-se 3.5 bilhões em 2017, sendo 64% via dispositivos móveis (CLARKE, 2013).

Sendo assim, de acordo com essa expansão, haverá um patamar em que tudo o que é possível estar, estará conectado, ampliando o conceito de Internet das Coisas para o de Internet de Todas as Coisas, segundo a Cisco (CISCO, 2016). Com essa quantidade de pessoas e dispositivos conectados, cerca de 44 trilhões de gigabytes serão gerados (EMC, 2014), os quais quando processados por sistemas inteligentes, ajudarão no surgimento de serviços de grande impacto no cotidiano de uma cidade (CLARKE, 2013).

Tal volume de dados, dobrará a cada dois anos até 2020 (EMC, 2014), principalmente devido ao crescimento do uso da Internet, smartphones e Redes Sociais; a queda do custo de equipamento para criar, capturar, administrar, proteger e armazenar informação; migração da TV analógica para a digital; crescimento dos dados gerados por máquinas, incluindo imagens de câmeras de segurança e metainformação (CLARKE, 2013).

Portanto, umas das principais problemáticas abordadas por Cidades Inteligentes é a de processamento de grande volume de dados, provenientes dos sensores instalados em tubulações de água, avenidas (para controle de congestionamentos), iluminações públicas; de câmeras de segurança; da análise de Redes Sociais, como o processamento de tweets (mensagem publicada ou trocada pelos utilizadores da rede social Twitter), etc., inserindo- se nesse contexto o conteúdo desse artigo.

Nesse contexto, este artigo apresenta uma aplicação para coleta e processamento de tweets, que obtém métricas relacionadas a e-Participação. As métricas são referentes as capitais dos estados brasileiros, focando as classificadas recentemente como Cidades Inteligentes (Connected Smartcities, 2015). Tais informações são exibidas numa aplicação web contendo um mapa que permite uma visão do nível de participação, no âmbito digital, entre o governo local e os cidadãos.

Além disso, são apresentadas duas aplicações para processamento de streams (fluxos) de tweets, que realizam sob eles Processamento de Linguagem Natural (NLP - Natural Processing Language), para obter a polaridade do sentimento do conteúdo do tweet. Com isso, estima-se em tempo real como a população está se sentindo em relação ao seu governo local.

A primeira das aplicações realiza processamento em lotes (batch), sendo o Apache Spark uma das ferramentas mais adequadas nesse sentido. A segunda, relaciona-se ao Processamento de Fluxo de Eventos (ESP - Event Stream Processing), processando eventos (acontecimentos do mundo real ou digital) na ordem em que eles chegam, o que é realizado com o Apache Storm.

Por fim, é feita uma breve revisão da literatura sobre as ferramentas citadas, analisando alguns requisitos importantes as aplicações desenvolvidas, sendo eles: Processamento de Grande Volume de Dados em Tempo Real, Tolerância a Falhas, Garantia de Processamento, Escalabilidade e Modelo de Programação (abstrações que influenciam o processo de desenvolvimento).

# II. Requisitos de uma aplicação de cidade inteligente

Devido a grande quantidade de dados existentes, conforme já mencionado anterior- mente, a capacidade de processar grandes volumes de dados pode ser um dos requisitos necessários a uma aplicação de Cidade Inteligente. Nesse contexto, por exemplo, pode ser importante observar duas métricas: o valor de (i) *throughput* e o de (ii) latência, sendo o primeiro termo referente a taxa de processamento e, o segundo, a variação do tempo entre um estímulo e resposta (KILLELEA, 2002).

Dependendo dos requisitos da aplicação, a latência talvez seja mais interessante de ser priorizada quando respostas a determinados eventos precisam ser no menor intervalo de tempo possível (menos de um segundo). Por outro lado, o valor de throughput, pode ser mais adequado se a aplicação necessitar processar grandes volumes de dados dentro de um valor de latência aceitável (no máximo alguns segundos) (MORAIS, 2015).

Outro requisito importante de ser analisado, é o de Tolerância a Falhas (necessário quando a aplicação está inserida em ambientes distribuídos, ou, de incertezas), o qual permite o funcionamento do sistema mesmo que uma falha (conceito explicado na seção 2.2) ocorra, o que pode ser obtido, por exemplo, por meio de replicação (COULOURIS et al., 2013). Além disso, pode existir a necessidade de atender o requisito de escalabilidade, através do qual é possível oferecer um serviço de alta qualidade conforme o aumento da demanda, adicionando novos recursos (escalabilidade horizontal), ou, melhorando os existentes (escalabilidade vertical) (SOMMERVILLE, 2011).

Por fim, algumas aplicações precisam do requisito de Processamento em Tempo Real, quem tem como uma de suas principais características um limite de tempo predeterminado para as respostas aos eventos do sistema (SOMMERVILLE, 2011). Após mencionar esses requisitos, é importante observar também o Modelo de Programação (ou, as abstrações) da plataforma que será́ utilizada no processo de desenvolvimento de uma aplicação, pois, ele (elas) pode (podem) impactá-los.

# III. e-participação

Uma das temáticas abordadas em Cidades Inteligentes, é a de promover novos meios para a participação do cidadão nas questões relacionadas a gestão da cidade (SAÉZ- MARTÍN; ROSSARIO; CABA-PEREZ, 2014). Dentre outras alternativas, as Redes Sociais são um dos principais meios onde essa interação pode ocorrer, posto que elas se definem por um conjunto de pessoas (ou, organizações), representando nós de uma rede, conectadas através de um conceito abstrato de relacionamento (MACIEL; ROQUE; GARCIA, 2009).

Sendo assim, tal ambiente virtual, tem proporcionado um novo espaço para que os cidadãos possam participar de processos de consulta e deliberação (exame e discussão de um assunto (PRIBERAM, 2016)), atuando com os governos como atores de processos de tomadas de decisão (MACIEL; ROQUE; GARCIA, 2009). Essa participação quando ocorre em ambientes virtuais, como Redes Sociais, aplicativos, Wiki, fórum, blogs, dentre outros (MAGALHÃES, 2015), define-se pelo conceito de e-Participação, dentro de outro mais abrangente que é o de Governo Eletrônico.

O Governo Eletrônico é caracterizado pelo uso de TICs pelo governo público, buscando prover melhores serviços, informações e conhecimento ao cidadão; facilitando o acesso ao processo político e incentivando a participação (MAGALHÃES, 2015). Ele pode ser dividido, principalmente, nos três campos seguintes: e-Administração, a respeito do funcionamento interno do poder publico; e-Governo, no tocante a entrega e fornecimento de serviços e informações qualificadas aos cidadãos; e-Democracia, relacionada a ampliação da participação da sociedade na tomada de decisão, sendo a e-Participação uma subárea desse último (MAGALHÃES, 2015).

A intenção da e-Participação é reforçar e renovar as interações entre o setor público, os políticos e cidadãos; tanto quanto possível no processo democrático (MAGALHÃES, 2015). Além disso, a e-Participação não pode ser avaliada somente por seus aspectos técnicos, mas também quanto a capacidade de incrementar a democracia (MAGALHÃES, 2015).

Um dos desafios nessa área é avaliar as ferramentas apoiadas pelas TICs, quanto as for- mas de engajamento existentes, como informação, consulta, ou, participação ativa (MAGA- LHÃES, 2015). Ainda, segundo citação contida em (MAGALHÃES, 2015), a e-Participação é um conjunto de várias tecnologias, medidas sociais e políticas, havendo devido a isso a necessidade de melhorar a compreensão das relações entre esses componentes e como suas respectivas práticas de avaliação podem ser aplicadas a e-Participação como um todo.

Com isso, para que os processos que a envolvem sejam eficazes é importante considerar os ambientes online e off-line, ou seja, incrementando os métodos tradicionais de participação (conferências, fóruns, conselhos, ouvidorias, audiências, consultas, reuniões, comitês, grupos de trabalho e mesas de negociação) com as possibilidades da e-Participação, estendendo-a ainda aos grupos desfavorecidos e desconectados (MAGALHÃES, 2015). Dito isso, no parágrafo seguinte, são referenciados alguns dos diferentes níveis de e-Participação.

No nível e-Informação, há um canal unidirecional, visando apenas fornecer informações de interesse cívico; no de e-Consulta, a comunicação é bidirecional, via coleta de opiniões e alternativas; no de e-Envolvimento busca-se garantir a compreensão e levar em consideração os anseios do cidadão; em e-Colaboração, a comunicação é bidirecional, e o cidadão participa ativamente no desenvolvimento de alternativas e identificação de melhores soluções; por fim, no de e-Empoderamento, a influência, controle e elaboração de políticas pelo e para o público é viabilizada (MAGALHÃES, 2015).

Explanados os diferentes níveis de e-Participação, podemos dizer que as Redes Sociais suportam não somente uma comunicação unidirecional, mas também bidirecional, a qual tem mais valia para as propostas da participação. A comunicação bidirecional (participação), no Twitter, pode ser identificada analisando algumas métricas, tais como: média do número de *tweets*, *retweets* (compartilhamento de um determinado *tweet*), comentá- rios realizados por usuários, réplicas a tweets, tempo de resposta e número de seguidores (SAÉZ-MARTÍN; ROSSARIO; CABA-PEREZ, 2014).

Ainda, tais métricas podem ser relacionadas com alguns dos seguintes indicadores propostos pela SNR (Social Network Ratio) para Redes Sociais: Atividade, ou, audiência estimada (número de seguidores); Tamanho, ou, esforço realizado pelo perfil para se comunicar (*tweets* por dia e tempo de resposta); Visibilidade, ou, número total de menções ao perfil; Interação (*retweets* e favoritos), ou, capacidade de impacto (viralização) da comunicação (*retweets*) (SAÉZ-MARTÍN; ROSSARIO; CABA-PEREZ, 2014).

# Iv. plataformas de processamento em tempo real

As plataformas de processamento em tempo real (RTC, Real Time Computing), compostas por hardware ou software, são sistemas que têm como um dos principais requisitos emitir respostas a eventos de acordo com um determinado deadline (limite de tempo). Sendo assim, a corretude da computação não depende apenas da corretude logica, mas também dos resultados serem produzidos de acordo com o deadline especificado (SHIN; RAMANATHAN, 1994), (SOMMERVILLE, 2011).

Normalmente, os resultados são obtidos através de processamentos realizados por um conjunto de tasks (tarefas) cooperativas, iniciadas por eventos do sistema. Os eventos são dependentes do ambiente no qual estão inseridos, podendo ocorrer periodicamente (de forma regular e previsível), ou, aperiodicamente (irregulares e imprevisíveis) (SHIN; RAMANATHAN, 1994), (SOMMERVILLE, 2011).

As tasks podem ter uma relação de interdependência e ao mesmo tempo nem todos os eventos que as originaram necessitarem de ser processados dentro de um deadline. Apesar disso, nenhuma task pode vir a comprometer o processamento de outra (SHIN; RAMANATHAN, 1994).

Com isso, os deadlines podem ser classificados em hard, firm, ou, soft. No primeiro caso, respectivamente, as respostas a todos os eventos devem necessariamente ocorrer dentro do deadline definido; no segundo, deadlines esporadicamente não atendidos são aceitos; no terceiro, deadlines não alcançados são permitidos frequentemente (SHIN; RA- MANATHAN, 1994). Além dessas categorias, há os sistemas com delay (atraso) introduzido entre o intervalo de tempo de estímulo e resposta, sendo consequentemente classificados como *Near Real Time* (NRT), ou seja, são sistemas de processamento em "quase tempo real".

Sendo assim, na categoria hard, é considerado como falha se o tempo estimado para um processamento não for atendido, e nas demais, a ocorrência disso resulta numa degradação (contínua de acordo com a quantidade de deadlines não atendidos) (SHIN; RAMANATHAN, 1994), (SOMMERVILLE, 2011).

Entende-se por falha quando o usuário não recebe algo esperado (por exemplo, deadline não atendido) devido a um erro de sistema. Sendo esse erro um estado errôneo resultante de um defeito (característica do sistema que pode resultar em erro). E, degradação, como decréscimo da qualidade (conceito subjetivo, de acordo com os requisitos da aplicação) do sistema (SOMMERVILLE, 2011).

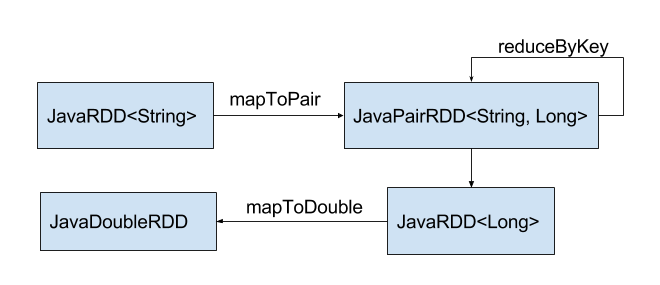
Além da caracterização por deadlines, tais plataformas podem se diferenciar quanto a abordagem de processamento, sendo uma delas é a ESP (*Event Stream Processing*). Em ESPs, algumas plataformas processam os streams continuadamente conforme são recebidos pelo sistema; paradigma conhecido como *One at Time*. Quando um evento é processado com sucesso, há a possibilidade de emitir uma notificação sobre isso, a qual é custosa devido ao fato de ser necessário rastreá-lo até o término de seu processamento (NARSUDE, 2015).

Outra alternativa, é a de realizar o processamento em *micro-batchs* (pequenos lotes) de eventos, tendo com uma das vantagens poder realizar operações em pequenos blocos de dados, em vez de individualmente, ao custo de introduzir um pequeno atraso no processo (NARSUDE, 2015). Exemplificando esses paradigmas, o Apache Spark fornece processamento em batch (lotes) e possui nativamente um módulo para micro-batchs, conhecido como Spark Streaming; por outro lado, o Apache Storm, realiza processamento ao modo *One at Time*.

v. Aplicação para processamento de tweets e coleta de métricas relacionadas a   
e-Participação

O primeiro passo no desenvolvimento dessa aplicação, foi decidir as contas (perfis) das quais os tweets seriam processados. Como, normalmente, as capitais dos estados têm maior concentração de pessoas, optou-se por fazer um levantamento dos perfis oficiais de suas respectivas prefeituras, para então posteriormente realizar o processamento dos tweets. Sendo assim, a Tab. 1 lista as contas relacionadas as prefeituras municipais das capitais dos vinte e sete estados brasileiros.

Em seguida, foram escolhidas quais métricas dessas contas seriam possíveis e importantes de coletar, tendo como referência (SAÉZ-MARTÍN; ROSSARIO; CABA-PEREZ, 2014). Sendo assim, selecionou-se os seguintes indicadores respectivos ao Twitter: média do número de tweets, seguidores, retweets (compartilhamento de um determinado tweet), comentários realizados por usuários, réplicas a tweets e tempo de resposta. As métricas referentes ao número de usuários acompanhando as listas (junções de timelines) dos perfis e o total delas existentes foram desconsideras, pois são relacionadas a contas diferentes das em questão.

De acordo com (SAÉZ-MARTÍN; ROSSARIO; CABA-PEREZ, 2014), através dessas métricas é possível obter indicadores relacionados a e-Participação. Alguns dos indicadores propostos pela SNR (Social Network Ratio) para Redes Sociais são: Atividade, ou, audiência estimada; Tamanho, ou, esforço realizado pelo perfil para se comunicar; Visibilidade, ou, número total de menções ao perfil; Interação, ou, capacidade de impacto (viralização) da comunicação (SAÉZ-MARTÍN; ROSSARIO; CABA-PEREZ, 2014).

Portando, pode-se mapear a métrica número de seguidores ao indicador Atividade; a média de menções realizadas para o de Visibilidade; o número de tweets, réplicas por dia e médias relacionadas ao tempo de resposta ao indicador Tamanho; e por último, as médias de retweets e favoritos ao de Interação. A aplicação para o processamento de tweets e coleta dessas métricas foi desenvolvida na linguagem de programação Java, devido a sua praticidade de uso, utilizando o framework para Web Services, Apache CXF (Apache Software Foundation, 2016a), para expor seus serviços através de uma REST API, permitindo integrá-la a aplicação web descrita **em**.

Nela, são coletados os 3.200 tweets mais recentes (se disponíveis) de cada conta, sendo tal quantidade limitada pela API do Twitter (acessada com o suporte da biblioteca Twitter4J (TWITTER4J, 2016)), a qual pode ser alcançada por no máximo 180 requisições, num intervalo de 15 minutos, com autenticação via conta de usuário (TWITTER, 2016). Esse é o mesmo limite para pesquisar as menções realizadas pelos cidadãos as contas das prefeituras, podendo ser coletados 100 tweets em cada requisição.

Durante a coleta dos tweets, os dados da conta são mapeados para um modelo contendo as informações relacionadas ao número de seguidores, tweets, localização, *username* e data de criação da conta; o mesmo procedimento é realizado para outro modelo, mapeando os dados de cada tweet, compondo data de criação, id do tweet de réplica e a qual usuário ele se refere, quantidade de retweets, favoritos, se é menção ou não, e o tempo de resposta calculado. Em seguida, os modelos são persistidos no banco de dados não relacional MongoDB (MongoDB, 2016).

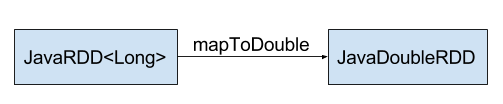
Os tweets coletados por essa aplicação são os mais recentes e anteriores a data 18/06/2016 (incluindo-a). Como essa coleta não foi realizada sob um stream de tweets, é realizado um processamento em batch desses, utilizando o Apache Spark, coletando as métricas relacionadas a e-Participação. Sendo assim, cada métrica é recuperada do banco de dados e mapeadas para um RDD de *doubles*, quando números, ou, de *strings*, no caso da data.

Figura 1: Diagrama do mapeamento entre um Resilient Distributed Dataset de Long para Double

Sendo assim, após recuperar as métricas, é possível mapeá-las para um RDD de doubles, por meio do qual são obtidos os valores relacionados as suas respectivas médias, medianas, variâncias, máximos, mínimos e desvios padrões. As informações sobre datas, como strings, são mapeadas para o valor 1, representando a ocorrência de um tweet nesse dia; compondo uma sequência de pares, que permite obter e mapear as quantidades de tweets por dia a um RDD de doubles. Tais processos de mapeamentos são ilustrados nas Fig. 1 e Fig. 2.

Figura 2: Diagrama do mapeamento entre um Resilient Distributed Dataset de Datas para suas respectivas frequências em Double

Por fim, os valores das métricas relacionadas a e-Participação são submetidos a uma Fusion Table (DIAS, 2016a), via a API do Google Fusion Tables, ocorrendo o mesmo para as referentes as polaridades de sentimentos, pro- cessados pelas aplicações explicadas **em**. Os valores dessa tabela são utilizados para criar o mapa contido na aplicação web. O código dessa aplicação está disponível no repositório localizado em (DIAS, 2016b), sendo "twitter-data-analysis" o nome do projeto.

Tabela I

Contas do Twitter relacionadas as prefeituras municipais das capitais dos vinte e sete estados brasileiros e quantidade de tweets processados para coleta das métricas relacionadas a e-Participação

|  |  |  |  |
| --- | --- | --- | --- |
| Estado | Capital | Conta no Twitter | Tweets processados |
| Acre | Rio Branco | PrefRioBranco | 3265 |
| Alagoas | Maceió | PrefMaceio | 3308 |
| Amapá | Macapá | PMMacapa | 3618 |
| Amazonas | Manaus | PrefManaus | 3230 |
| Bahia | Salvador | agecomsalvador | 1231 |
| Distrito Federal | Brasília | Gov\_DF | 3559 |
| Ceará | Fortaleza | prefeiturapmf | 3678 |
| Espírito Santo | Vitoria | VitoriaOnLine | 3249 |
| Goiás | Goiânia | PrefeituraGy | 4053 |
| Maranhão | São Luís | PrefeituraSL | 3720 |
| Mato Grosso | Cuiabá́ | prefeitura\_CBA | 3211 |
| Mato Grosso do Sul | Campo Grande | cgnoticias | 3200 |
| Minas Gerais | Belo Horizonte | prefeiturabh | 3623 |
| Paraná | Curitiba | Curitiba\_PMC | 4951 |
| Paraíba | João Pessoa | pmjponline | 3677 |
| Pará | Belém | prefeiturabelem | 1131 |
| Pernambuco | Recife | prefrecife | 3725 |
| Piauí | Teresina | prefeitura\_the | 3392 |
| Rio Grande do Norte | Natal | NatalPrefeitura | 3360 |
| Rio Grande do Sul | Porto Alegre | Prefeitura\_POA | 3529 |
| Rio de Janeiro | Rio de Janeiro | Prefeitura\_Rio | 6387 |
| Rondônia | Porto Velho | prefeitura\_pvh | 2805 |
| Roraima | Boa Vista | PrefeituraBV | 3201 |
| Santa Catarina | Florianópolis | scflorianopolis | 3448 |
| Sergipe | Aracaju | PrefeituraAracaju | 3423 |
| São Paulo | São Paulo | prefsp | 4330 |
| Tocantins | Palmas | cidadepalmasy | 3574 |
| **Total** |  |  | 93878 |

v. Aplicação para processamento de stream de tweets, utilizando   
spark streaming

A aplicação que utiliza o Spark Streaming, foi desenvolvida na linguagem de programação Java. No início de sua execução, é criado um contexto de stream no qual é definindo o cluster onde ela será́ executada e o intervalo de criação de cada RDD. Tais Resilient Distributed Datasets, são compostos ordenadamente em sequência, formando a abstração conhecida como DStream. No nosso caso, cada RDD é criado após 30000ms; sendo compostos pelos tweets coletados filtrando os nomes das contas das prefeituras no Twitter.

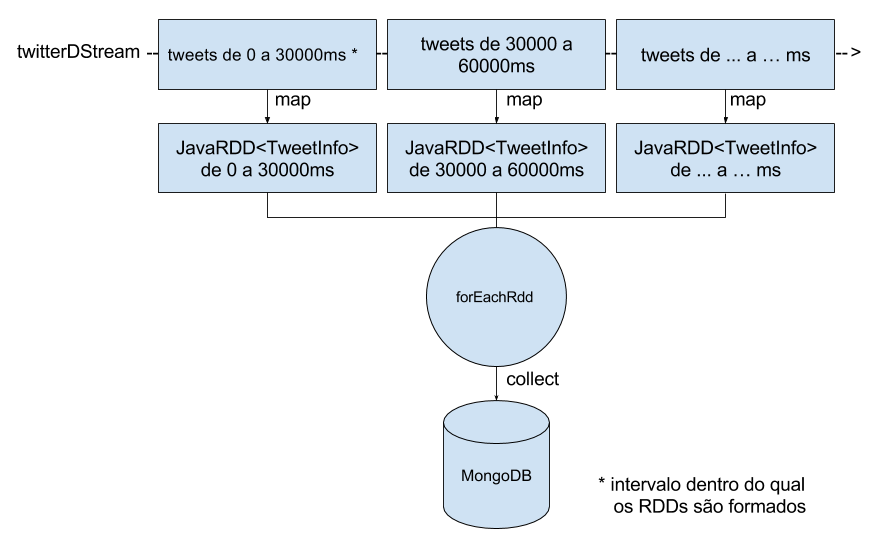
O processo mencionado anteriormente, começa após a inicialização do contexto de stream. Durante a coleta dos streams de tweets (eventos), cada RDD é mapeado para o modelo da aplicação, através de uma transformação map. Na sequência, as ações forEachRDD e collect são executadas para inserir os tweets processados no banco de dados não relacional MongoDB, conforme ilustrado na Fig. 3.

Figura 3: Fluxo do processamento de dados da aplicação utilizando o Spark Streaming

A análise das polaridades dos sentimentos do conteúdo contido nos tweets ocorre durante a última parte do mapeamento, sendo antes disso realizados alguns processamentos para facilitar e viabilizar esse procedimento. Portanto, primeiramente, os textos contidos nos tweets são extraídos e formatados para minúsculo. Em seguida, todas as menções são identificadas pelo padrão em que ocorrem no Twitter (@displayname, nome exibido para os demais usuários da Rede Social) e removidas, assim como as referências a endereços de sites.

No Twitter, quando um tweet é compartilhado ele é marcado com a notação "RT", abreviação de retweet, a qual também é removida do texto em processamento. Além disso, alguns símbolos são removidos, tais como: ., ., , %, #, etc. Sendo esse o conjunto de processamentos realizados para "limpar" inicialmente o texto, após o qual se inicia o Processamento de Linguagem Natural.

Com esse proposito, usou-se a biblioteca OpenNLP (Apache Software Foundation, 2016e) para a tokenização dos tweets. Após a obtenção dos tokens, outros processamentos foram necessários para melhorar o desempenho da fase seguinte, como a substituição das palavras normalmente abreviadas (vc - você̂, msm - mesmo, pq - porque, q - que, n - não, etc) e de expressões (sqn - só́ que não, kkk, hahaha, rsrsrs, para situações comumente engraçadas) para seus respectivos formatos formais, utilizando dicionários previamente construídos. Além disso, foram removidos os tokens contendo "stopwords"(palavras vazias), termo utilizado para as palavras comuns de um certo idioma (LESKOVEC; RAJARAMAN; ULLMAN, 2016).

Após a obtenção dos tokens, foram atribuídas a eles tags referentes as suas respectivas classes gramaticais, e, por fim, associadas a uma part-of-speech para cada tweet, usando os tokens e tags obtidas, finalizando a parte do Processamento de Linguagem Natural. Seguindo então, para a última etapa, que é a da análise de polaridade dos sentimentos, tendo como base para isso os adjetivos presentes em cada tweet.

A análise das polaridades dos sentimentos foi realizada com o suporte do Sentilex (versão 1), que é um léxico de sentimentos para o Português, constituído de 6.321 lemas adjetivais (por convenção, na forma masculina singular) e 25.406 formas flexionadas, contendo como um de seus atributos a polaridade do adjetivo. As polaridades são classificadas em positivo (67% de precisão), negativo (82% de precisão), ou, neutro (45% de precisão), possibilitando estimar o sentimento expresso por um determinado texto (SILVA et al., 2016).

Os sentimentos são estimados contabilizando as polaridades presentes em cada tweet e o sentimento expresso pelos emotions (se houver), assim sendo, por exemplo, o emotion "(:"incrementa 1 para a polaridade positiva, e 1 para a negativa caso seja ":(". Também, considera-se a presença de advérbios de negação, os quais modificam o significado do verbo, adjetivo e de outros advérbios (PRIBERAM, 2016), alterando consequentemente a polaridade. Por fim, é realizada uma simples normatização com os somatórios das polaridades positivas e negativas, seguindo o seguinte modelo:

score = (Σ positivo - Σ negativo) ÷ (Σ positivo + Σ negativo)

Caso o score seja menor do que zero, o tweet é classificado com polaridade negativa, se o seu complemento for maior do que 0.5, tem-se polaridade positiva e se igual a zero, neutra. Sendo assim, as informações sobre a polaridade (sentimento) do tweet, seu id e suas respectivas menções (recuperadas do tweet original) são armazenadas no banco de dados não relacional MongoDB. As polaridades são exibidas no mapa da aplicação web. O código dessa aplicação está disponível no repositório localizado em (DIAS, 2016b), sendo "spark-data-analysis"o nome do projeto.

vI. Aplicação para processamento de stream de tweets, utilizando   
spark streaming

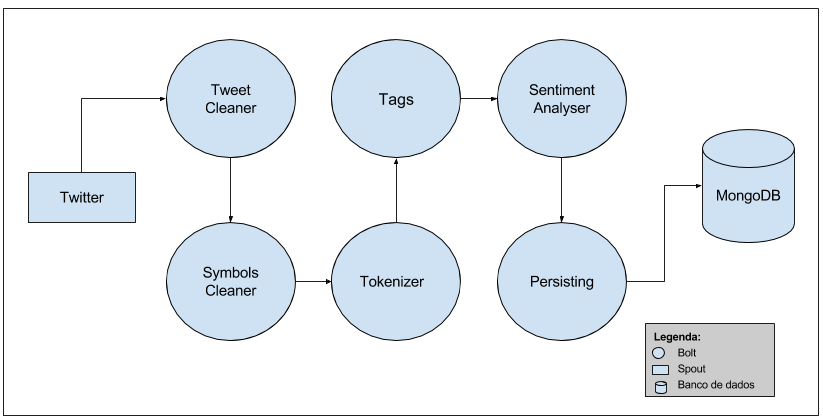
A aplicação que utiliza o Storm para o processamento de stream de tweets, foi desenvolvida utilizando a linguagem de programação Java. Nela é construída uma topologia, ilustrada na Fig. 4, composta por um Spout (classe Twitter), responsável pela conexão ao Twitter e coleta dos tweets, tendo como filtro o nome das contas das prefeituras no Twitter, utilizando a biblioteca Twitter4J.

Figura 16: Topologia da aplicação utilizando Storm

Em sequência, há seis bolts, responsáveis pelo processamento (o mesmo realizado pela aplicação Spark) dos tweets coletados. O primeiro deles, remove as menções e urls contidas no tweet, após isso os símbolos existentes são removidos pelo bolt Symbols Cleaner.

Continuando, a parte do Processamento de Linguagem Natural é realizado pelos bolts Tokenizer e Tags. Na finalização do processo, o bolt SentimentAnalyser calcula a polarização de cada tweet, emitindo-os para o bolt Persisting, responsável pelo armazená-los no banco de dados não relacional MongoDB. O código dessa aplicação está disponível no repositório localizado em (DIAS, 2016b), sendo "storm-data-analysis" o nome do projeto

vII. Revisão da literatura

Quanto ao requisito de Processamento de Grandes Volumes de Dados em Tempo Real, de acordo com a referência (DAS, 2016), o Apache Spark pode processar até 670k de registros por segundo e por nó, enquanto que o Storm 155k. Portanto, ambas as ferramentas são capazes de processar grande fluxo de dados.

No entanto, elas se diferenciam no requisito de processamento em tempo real. O Spark tem latência de 0.52s (ZAHARIA et al., 2012b), e devido a esse delay de até alguns segundos, é considerado uma ferramenta de Near Real Time. O Storm, por outro lado, tem como latência de 1100ms (ZAHARIA et al., 2013), (ZAHARIA et al., 2012b), sendo por isso um sistema com processamento em Real Time.

Quanto menor o valor de latência, mais rápido se obtém resposta a um dado evento. Além disso, no quesito de processamento em tempo real, também é importante avaliar o valor de throughput, o qual no Spark é maior em comparação ao do Storm (ZAHARIA et al., 2013), favorecendo menor tempo de processamento.

Como base no que foi dito anteriormente, nesse quesito o Spark é mais adequado as aplicações desenvolvidas, pois elas estão inseridas num contexto em que a propriedade de throughput tem maior relevância. Isso, devido a quantidade de tweets processados, tanto para análise das polaridades de sentimentos, com o módulo Spark Streaming, quanto para o processamento das métricas relacionadas a e-Participação.

3.2.2 Tolerância a FalhasDiscretized Streams (DStreams) é um modelo de programação para processamento de streams distribuídas, utilizado pelo Spark Streaming, capaz de fornecer Tolerância a53

Falhas, através do método parallel recovery (ZAHARIA et al., 2012c). Nesse modelo, a tole- rância a falhas é implementada através do conceito lineage, o que permite as informações serem recuperadas paralelamente (ZAHARIA et al., 2012a).

O mecanismo de recuperação via lineage é definido por um grafo acíclico dirigido (DAG), por meio do qual RDDs e DStreams rastreiam, ao nível das partições RDDs, suas respectivas dependências e operações realizadas sob elas (ZAHARIA et al., 2012a). Sendo assim, os RDDs e DStreams conseguem "saber"como foram construídos. Podendo, consequentemente, cada nó do cluster reconstruí-lo paralela e eficientemente em caso de falhas. Especificamente, o processo de recuperação é realizado computando novamente uma determinada partição RDD, reexecutando as tasks que a originaram (ZAHARIA et al., 2012a).

Visando prevenir infinitas re-computações, também são realizados checkpoints num determinado espaço de tempo, com replicações assíncronas de RDDs. Tal procedimento, não é necessário para todo o conjunto de dados, pois, como já mencionado, a recuperação executada por nós em paralelo é realizada com demasiada eficiência (ZAHARIA et al., 2012a).

Todo o processo descrito até aqui é confiável quando a fonte dos dados pode ser lida novamente, no caso do Spark Streaming, é necessário haver alguma fonte externa para replicação dos dados (Apache Software Foundation, 2016g). Além disso, se o Driver for finalizado, devido ao fato dele manter o contexto da aplicação, todo o conteúdo em memória dos executors é perdido (Apache Software Foundation, 2016g).

No Storm, a Tolerância a Falhas é aplicada a cada um de seus componentes. Por exemplo, se o worker falha, ele é reinicializado pelo Supervisor no próprio nó (JAIN; NALYA, 2014). Caso o nó esteja indisponível, ou, falhando continuadamente, suas tasks são então atribuídas a outro disponível no cluster, via Nimbus (HART; BHATNAGAR, 2015).

O Nimbus e os supervisors, armazenam seus estados no Zookeeper, podendo ser reini- cializados sem perdê-los em caso de falha, se houver falha no Zookeeper, outra instância pode ser "eleita"para o lugar dele(JAIN; NALYA, 2014). Caso algum supervisor falhe, seus workers são reatribuídos pelo Nimbus a outro supervisor, no entanto, ficando impedido de receberem novas tuplas (HART; BHATNAGAR, 2015).

Por sua vez, o Nimbus, dentre suas atribuições, é responsável também por reinicializar as tasks caso uma delas falhe, e se o mesmo acontecer com ele próprio, as tasks em execução não são afetadas, mas novas topologias são impedidas de serem submetidas ao54

cluster (JAIN; NALYA, 2014), assim como reatribuições (HART; BHATNAGAR, 2015).

O modelo de recuperação de falhas do Spark pode ser uma boa escolha caso a aplica- ção permita perda de dados, em prol de eficiência (ZAHARIA et al., 2012c). Caso contrário, é necessário configurar uma fonte externa para replicação dos dados que estão sendo pro- cessados. Em contraste, a arquitetura do Storm, busca possibilitar que seus componentes falhem havendo pouco prejuízo para a aplicação, ainda assim a replicação pode adicionar mais uma camada de confiabilidade.

Levando em consideração o exposto acima, a aplicação desenvolvida para o processa- mento de métricas, realiza-o em batch, podendo ter acesso a fonte de dados para coletá-los novamente, em caso de falha. Além disso, a eficiência foi priorizada, não havendo prejuízo ao utilizar o Spark. Para aplicação que realiza análise de polaridade dos sentimentos, pro- cessando stream de tweets, é mais interessante o suporte a Tolerância a Falhas do Storm, principalmente devido longo tempo de execução dela.

3.2.2.1 Garantia de processamento

Em adição a subseção anterior, o Spark Streaming e Storm possuem formas diferentes de garantir o processamento de um evento. No Spark Streaming há a garantia de que todo evento será processado exatamente uma vez (Exactly Once), sem perdas, ou, duplicadas, via parallel recovery, levando em consideração as observações já mencionadas (Apache Software Foundation, 2016g).

No Storm, por outro lado, não há suporte ao modelo Exactly Once, mas sim aos At Least Once e At Most Once. Em ordem, a primeira opção permite que o processamento seja realizado no mínimo uma vez, rastreando se o (a) evento (tupla) foi processado(a) ou não, através de seus IDs e mensagens informando "ack"(tupla processada), podendo haver duplicatas; o oposto ocorre na segunda alternativa, em que perdas são aceitas, processando os eventos no máximo uma vez (Apache Software Foundation, 2016h).

A proposta do Spark Streaming, nesse aspecto, é mais interessante para o processa- mento de stream de tweets, por garantir que a informação será processada uma única vez, através do modelo Exactly Once.

3.2.3 EscalabilidadeQuanto ao requisito de escalabilidade, ambos suportam clusterização, portando são escaláveis horizontalmente, sendo o maior cluster Spark conhecido composto por 8.00055

nós (Apache Software Foundation, 2016g). E, embora não tenham sido encontradas fontes confiáveis divulgando informações sobre o maior cluster Storm existente, sabe-se que ele é capaz de processar um milhão de mensagens com tamanho de 100 byte, por segundo e por nó (Apache Software Foundation, 2016h).

3.2.4 Modelo de programação

Conforme exposto no Cap. 2, no Spark os RDDs são conjuntos de dados que podem ser manipulados de forma distribuída, sendo possível realizar operações sob eles, gerando novos RDDs, através de transformações, ou, computando-os por meio das ações. O módulo Streaming, utiliza essa unidade como base da abstração DStream, conjunto de RDDs representando um stream. Tais conceitos do modelo de programação do Spark podem ser mais difíceis de abstrair, tendo uma classificação de baixo nível, se comparados ao do Storm.

Por outro lado, numa perspectiva alto nível, o Storm propõe o conceito de Topolo- gia, composta por bolts e spouts. Os spouts são responsáveis pela entrada dos streams (conjuntos de tuplas) da aplicação, processados posteriormente pelos bolts. Devido a essas abstrações, consequentemente, pode haver maior facilidade em programar utilizando o Storm do que o Spark.

1. M. S. M. A. Notare, Latin America Transactions, IEEE (Revista IEEE America Latina) Editor-in-Chief, IEEE South Brazil, FAERO Technology University, [mirela@ieee.org](mailto:mirela@ieee.org) [↑](#footnote-ref-1)