A Platform for Measuring e-Participation in Smart Cities: A Case Study with Brazilian Capitals

F. C. A. Dias and N. A. A. Cacho

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 in questions addressed by your respective local government, which is essential for the development of Smart Cities and known as *eParticipation*. This paper describes a platform that collects and processes tweets (Social Network Twitter posts) to produce metrics related to e-Participation. In order to assess the proposed platform, we have instantiated the platform using two data processing tools: Apache Spark and Apache Storm.**

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

I. INTRODUÇÃO

Atualmente, o crescimento populacional tem sido uma das fontes de preocupação dos gestores da infraestrutura e dos serviços de uma cidade [1]. À medida que crescem as cidades, aumenta a complexidade e os desafios de gestão para as autoridades governamentais em identificar problemas relativos ao abastecimento de água, coleta de lixo local, sistema de gestão de tráfego urbano, saúde, educação, segurança pública, economia, meio ambiente e turismo. No sentido de facilitar a identificação de tais problemas, as Tecnologias de Informação e Comunicação (TICs) vem sendo utilizadas para construir o conceito de Cidades Inteligentes (CI) [3].

A implementação de iniciativas de Cidades Inteligentes geralmente demanda a utilização de uma variedade de sensores e aplicações pervasivas que estão conectadas e podem interagir entre si para criarem ambientes monitoráveis e que podem ser ajustados em tempo real [29]. Assim, em muitos casos o potencial de Cidades Inteligentes depende da densidade e integração dos sensores e suas aplicações. Exemplos de aplicações em Cidades Inteligentes incluem: implantação de sensores de vibração para monitorar o fluxo de veículos e a integridade do pavimento [30], uso de sensores para prever condições de transito e otimizar a iluminação pública em Lion, na França [31], etc.

Tais aplicações necessitam coletar simultaneamente dados de diferentes fontes sensoriais o que requer um cenário de sensoriamento de múltiplas partes (multiparte). A heterogeneidade de fontes sensoriais é um desafio para sensoriamento multiparte, uma vez que diferentes tecnologias impõem tarefas adicionais para lidar com aspectos específicos, como protocolos de conectividade, sincronização de sinais, e agregação de dados na tecnologia do sistema. Além disso, os sensores estão distribuídos sobre a cidade inteligente, requerendo convergência de conectividade para conexões simultâneas de dispositivos anexados a diferentes redes. No contexto das cidades brasileiras, todos estes desafios se somam à questão financeira, dado que a implantação de uma ampla rede de sensoriamento demanda altos investimentos que a maioria das cidades brasileiras não possuem.

Diante disso, algumas pesquisas [27, 28] sugerem usar as redes sociais para identificar de maneira instantânea a percepção dos moradores e visitantes sobre uma determinada cidade. Segundo [27], o uso de sensores pode ser útil para identificar "o que" está ocorrendo, mas não é capaz de identificar "por que" e "como" tal incidente ocorre. Neste caso, [27] sugere utilizar as redes sociais para capturar a percepção humana sobre os eventos que acontecem nas cidades. Tal sugestão se baseia no fato de que a percepção sobre os eventos de uma cidade são geralmente descritos através de comentários nas redes sociais. Por exemplo, é comum observar comentários repetitivos reclamando sobre o trânsito quando da ocorrência de congestionamentos. A identificação de tais demandas por meio das redes sociais caracteriza o conceito de *e-Participação* [10] onde os cidadãos participam de processos de consulta e deliberação, atuando com os governos como atores de processos de tomadas de decisão [11].

Diferentemente da utilização de sensores, o uso das redes sociais como fonte de informação não demanda a instalação e manutenção de uma ampla infraestrutura de TI espalhada pela cidade. É necessário apenas um computador conectado a internet que faça uso da API fornecida pela rede social utilizada. Apesar destas facilidades, a utilização de redes sociais no contexto de Cidades Inteligentes apresenta vários desafios. O primeiro é ser capaz de processar em tempo real todas as postagens enviadas sobre assuntos diversos. O segundo desafio é ser capaz de interpretar tais postagens de modo que o sentimento expresso pelo texto da postagem possa ser utilizado para identificar a ocorrência de eventos. O terceiro desafio é armazenar tais dados de modo que eles possam gerar visualizações adequadas para o interesse dos gestores.

No sentido de tratar tais desafios este artigo possui como objetivos: (i) apresentar uma plataforma que utiliza as redes sociais como fonte de dados para apoiar as decisões dos gestores no contexto de uma cidade inteligente, (ii) instanciar a plataforma proposta nas duas principais plataformas de processamento em tempo real disponíveis (*Apache Spark* e *Apache Storm*), (iii) coletar métricas sobre as 27 capitais de

F. C. A. Dias, University of São Paulo, São Paulo, Brasil, felipecavazotto@usp.br

N. A. A. Cacho, Department of Informatics and Applied Mathematics, Federal University of Rio Grande do Norte, Natal, Brasil, neliocacho@dimap.ufrn.br

estado do Brasil, (iv) avaliar a plataforma proposta segundo critérios como *throughput* e *latência*.

II. REQUISITOS DE UMA APLICAÇÃO DE CIDADE INTELIGENTE

Devido a quantidade de dados que são gerados diariamente pelos moradores de uma cidade, a capacidade de processá-los pode ser um dos principais requisitos necessários para o desenvolvimento de uma aplicação de Cidade Inteligente. Tais requisitos envolvem questões, como: o valor de (i) *throughput,* (ii) *latência*, (iii) *tolerância a falhas* e (iv) *escalabilidade.*

O *throughput* se refere a taxa de processamento e a *latência* está relacionada a variação do tempo entre um estímulo e resposta [6]. Dependendo dos requisitos da aplicação, a latência talvez seja 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 (alguns segundos) [7].

Outro requisito importante de ser analisado, é o de *Tolerância a Falhas*, o qual permite o funcionamento do sistema mesmo que uma falha ocorra [8]. Além disso, atendendo questões de *escalabilidade*, é possível oferecer um serviço de alta qualidade conforme o aumento da demanda, por meio de escalabilidade horizontal ou vertical [9].

III. PLATAFORMAS DE PROCESSAMENTO EM TEMPO REAL

Plataformas de processamento em tempo real (RTC, *Real Time Computing*) vem sendo utilizadas em projetos de Cidades Inteligentes para atender aos requisitos mencionados na Seção II. Nesses sistemas, requer-se respostas a eventos de acordo com um determinado *deadline*. Sendo assim, a corretude da computação não depende apenas da corretude lógica, mas também do *deadline* especificado [13, 14].

Normalmente, os resultados são obtidos por um conjunto de *tasks* cooperativas, iniciadas por eventos do sistema. Os eventos são dependentes do ambiente no qual estão inseridos, podendo ocorrer periodicamente, ou, aperiodicamente [13,14]. As *tasks* podem ter uma relação de interdependência e ao mesmo tempo nem todos os eventos que as originaram necessitam ser processados dentro de um *deadline*. Apesar disso, nenhuma *task* pode vir a comprometer o processamento de outra [13].

Com isso, os *deadlines* podem ser classificados em *hard*, *firm*, ou, *soft*. No primeiro caso, respectivamente, as respostas aos eventos devem ocorrer dentro do *deadline* definido; no segundo, *deadlines* esporadicamente não atendidos são aceitos; no terceiro, os não alcançados são permitidos frequentemente [13]. Além disso, há também os sistemas com *delay* introduzido, classificados como *Near Real Time* (NRT). Assim, na categoria *hard*, é considerado como falha se o *deadline* de um processamento não for atendido, e nas demais, uma degradação contínua [13, 14].

Tais plataformas, podem também se diferenciarem quanto a abordagem de processamento, sendo uma delas a ESP (*Event Stream Processing*). Em ESPs, algumas plataformas processam os streams conforme são recebidos pelo sistema (paradigma *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 [15].

Outra alternativa, é a de realizar o processamento em *micro-batchs* de eventos, tendo como uma das vantagens realizar operações em pequenos blocos de dados, em vez de individualmente, ao custo de introduzir um pequeno atraso no processo [15]. Dois exemplos destes paradigmas são o *Apache Spark* e o *Apache Storm*. O *Apache Spark* fornece processamento em *batch* e tem nativamente um módulo para *micro-batchs* (o *Spark Streaming*) por outro lado, o *Apache Storm*, realiza processamento ao modo *One at Time*.

IV. E-PARTICIPAÇÃO

Uma das temáticas abordadas em Cidades Inteligentes, é a de promover novos meios para a participação do cidadão nas questões de gestão da cidade [3]. Nisso, as Redes Sociais são meios viabilizadores a interação, posto que elas se definem por um conjunto de pessoas representando nós de uma rede, conectadas por um conceito abstrato de relacionamento [11].

Sendo assim, esses ambientes proporcionam novos espaços para que os cidadãos participem de processos de consulta e deliberação, atuando com os governos como atores de processos de tomadas de decisão [11]. Tal interação define o conceito *e-Participação* [10], situado em Governo Eletrônico.

A intenção da *e-Participação* é reforçar e renovar as interações entre o setor público, os políticos e cidadãos, incrementando as formas de participação e democracia já existentes, estendendo-a a grupos desfavorecidos e desconectados; tanto quanto possível no processo democrático [10]. Além disso, a *e-Participação* é dividida em níveis.

No nível *e-Informação*, há um canal unidirecional, visando 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 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 [10].

Portanto, as Redes Sociais suportam as comunicações unidirecionais e bidirecionais (de mais valia para as propostas da participação). A comunicação bidirecional no *Twitter*, pode ser identificada analisando algumas métricas, tais como: média do número de *tweets*, *retweets*, comentários, réplicas a *tweets*, tempo de resposta e número de seguidores [3].

Ainda, tais métricas podem ser relacionadas com alguns dos seguintes indicadores propostos pela SNR (*Social Network Ratio* [3]) para Redes Sociais: Atividade, ou,

audiência estimada (número de seguidores); Tamanho, ou, esforço realizado 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*) [3]. Algumas destas métricas foram implementadas pela plataforma proposta e serão descritas na Seção V.

V. PLATAFORMA DE E-PARTICIPAÇÃO

Como descrito na seção anterior, existe uma grande disponibilidade de dados que permitem implantar abordagens de gestão por meio da utilização da *e-Participação*. Neste sentido, esta seção descreve uma plataforma de *e-Participação* que permite os gestores públicos identificar demandas e avaliar o nível de *e-Participação* das suas cidades. A plataforma proposta utiliza as redes sociais para coletar e analisar a percepção dos moradores e visitantes, armazenar em bancos de dados e visualizar os resultados de diversas formas.

V.I ARQUITETURA

A arquitetura da plataforma de *e-Participação* é composta por quatro componentes: (i) *abstração para infraestrutura de tempo real*, (ii) *processamento*, (iii) *armazenamento* e (iv) *dashboard*. Todos estes componentes foram desenvolvidos na linguagem de programação *Java*, utilizando o framework para *Web Services*, *Apache CXF*, para expor seus serviços através de uma *REST API*, acessando a *API* do *Twitter* via a biblioteca *Twitter4J*.

O componente de abstração permite isolar as complexidades no manuseio das plataformas de processamento de tempo real utilizadas pela plataforma proposta: *Spark Streaming* e *Apache Storm*. Cada uma destas plataformas será utilizada para desenvolver um tipo de aplicação que será descrito na seção V.III e V.IV. O processo de coleta dos dados é realizado pelo componente de *processamento*. Este componente é responsável por coletar e processar os dados das contas selecionadas que 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 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*.

Por fim, os valores das métricas relacionadas a *e-Participação* são submetidos a uma *Fusion Table* [19], via a *API* do *Google Fusion Tables*, ocorrendo o mesmo para os referentes as polaridades de sentimentos, processados pelas aplicações descritas na Seção V.III e V.IV. Os valores dessa tabela são utilizados para criar o mapa contido na aplicação *web* suportada pelo componente *dashboard*.

V.II MÉTRICA

Para avaliar o nível de *e-Participação* das 27 capitais de estados brasileiros foram escolhidas as seguintes métricas (como base em [3]) referente à conta do *Twitter* dos perfis listados na Tab. I: média do número de *tweets*, seguidores, *retweets*, comentários realizados por usuários, réplicas a *tweets* e tempo de resposta. De acordo com [3], 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 [3].

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.

TABELA I

CONTAS DO TWITTER RELACIONADAS ÀS PREFEITURAS MUNICIPAIS DAS CAPITAIS DOS VINTE E SETE ESTADOS BRASILERISO E TOTAL 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	Cuiabá	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.III. PLATAFORMA USANDO SPARK

Conforme descrito, a plataforma proposta suporta as duas principais plataformas de processamento em tempo real disponíveis. Nesta seção será descrito como a plataforma proposta foi configurada para utilizar a plataforma *Apache Spark*. Os *tweets* coletados por essa aplicação foram coletados durante 7 dias a contar do dia 18/05/2016.

No início da execução da aplicação proposta para avaliar a plataforma (disponível em [20]), é criado um contexto de *stream* no qual é definindo o *cluster* onde ela será executada e o intervalo de criação de cada *Resilient Distributed Datasets* (RDD). Tais *RDDs*, são compostos ordenadamente em sequência, formando a abstração conhecida como *DStream*. No caso desta aplicação, cada RDD é criado após 30000ms; sendo compostos pelos *tweets* coletados filtrando os nomes das contas das prefeituras no *Twitter*. Durante a coleta dos *streams* de *tweets*, cada RDD é mapeado para o modelo da aplicação, através de uma transformação *map*. Na sequência,

as ações *forEachRDD* e *collect* são executadas para armazenar os *tweets* processados.

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. Primeiramente, os textos contidos nos *tweets* são extraídos e formatados para caixa baixa, as menções, *retweets*, são identificadas e removidas, assim como as referências a endereços de sites e símbolos. Em seguida a essa mineração, inicia-se o Processamento de Linguagem Natural.

Com esse proposito, usou-se a biblioteca *OpenNLP* [21] para a *tokenização* dos *tweets*. Após isso, o processamento foi otimizado, substituindo as palavras e expressões abreviadas para seus respectivos formatos formais, utilizando dicionários previamente construídos e removendo os *tokens* contendo *stopwords* [22].

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 [23].

Os sentimentos são estimados contabilizando as polaridades presentes em cada *tweet* e o sentimento expresso pelos *emotions*. Também, considera-se a presença de advérbios de negação, os quais modificam o significado do verbo, adjetivo e de outros advérbios, alterando consequentemente a polaridade. Por fim, é realizada uma simples normatização com os somatórios das polaridades positivas e negativas, seguindo o seguinte modelo:

(I) $score = (\Sigma \text{ positivo} - \Sigma \text{ negativo}) \div (\Sigma \text{ positivo} + \Sigma \text{ negativo})$

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 do *tweet*, seu *id* e suas respectivas menções são armazenadas no banco de dados não relacional *MongoDB*. As polaridades são exibidas no componente *dashboard*.

V.IV PLATAFORMA USANDO STORM

A mesma aplicação desenvolvida para o *Apache Spark* foi implementada (disponível em [20]) para o *Apache Storm*. Para o *Apache Storm* foi construída por uma topologia, composta por um *Spout* responsável pela conexão ao *Twitter* e coleta dos *tweets*, tendo como filtro o nome das contas das prefeituras. 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*, que os armazena no componente de armazenamento.

VI. RESULTADOS

O gráfico exibido na Fig. 1 resume os valores das métricas coletadas (disponíveis em [20]), nele cada métrica é atribuída uma pontuação, equivalente a classificação contida em [20], ou seja, o 27º lugar representa 27 pontos; quanto menor o número de pontos, melhor a pontuação final. A pontuação final é composta pelo somatório dos pontos de todas as métricas.

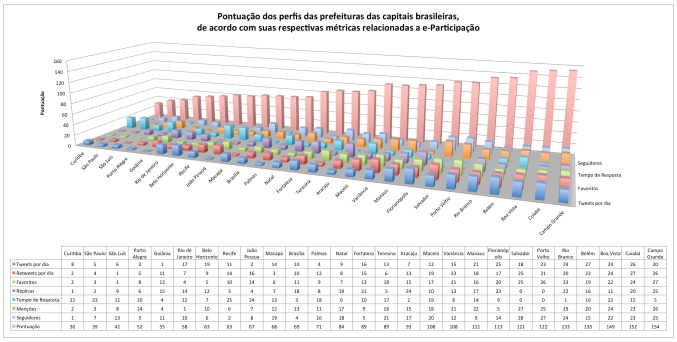
Visando comparar essa classificação, podemos citar o trabalho realizado em [5], que classificou algumas das cidades brasileiras como Cidades Inteligentes levando em consideração 70 indicadores, sendo nenhum deles relacionados a *e-Participação*. Com esse fim, foi construída a Tab. II, contendo apenas a colocação das cidades analisadas por esse artigo. Como esperado, analisando ambas as classificações, nem todas as colocações são correspondentes.

TABELA II

COLOCAÇÕES [1] DAS CIDADES INTELIGENTES BRASILEIRAS [2] EM RELAÇÃO ÀS DE *e-Participação*

UF	Capital	Cidades Inteligentes [3]	*e-Participação* [4]
RJ	Rio de Janeiro	1º	6º
SP	São Paulo	2º	2º
MG	Belo Horizonte	3º	7º
DF	Brasília	4º	10º
PR	Curitiba	5º	1º
ES	Vitória	7º	16º
SC	Florianópolis	8º	18º
RS	Porto Alegre	9º	4º
PE	Recife	10º	7º
CE	Fortaleza	18º	13º
GO	Goiânia	24º	5º
PB	João Pessoa	29º	8º
BA	Salvador	31º	-
SE	Aracaju	33º	14º
PI	Teresina	35º	13º

[1] As cidades ausentes não estão classificadas como inteligentes ou não são capitais.
[2] Ranking geral composto por 50 cidades.
[3] Colocação como Cidade Inteligente.
[4] Colocação em *e-Participação*. O hífen indica desclassificação por ausência de uma das métricas analisadas.



	Curitiba	São Paulo	Rio de Janeiro	Porto Alegre	Goiânia	Belo Horizonte	Recife	João Pessoa	Maceió	Brasília	Palmas	Natal	Fortaleza	Teresina	Aracaju	Macapá	Florianópolis	Salvador	Porto Velho	Rio Branco	Belém	Boa Vista	Cuiabá	Campo Grande
Tweets por dia	8	5	6	3	1	17	10	11	2	16	10	4	9	16	13	7	12	15	21	25	18	23	24	27	26	26	20
Retweets por dia	2	4	1	5	11	7	9	14	16	3	10	12	8	15	6	13	19	23	18	17	35	21	28	22	24	27	26
Favoritos	2	3	1	8	12	4	5	10	14	6	11	9	7	13	18	15	17	21	16	20	25	26	23	19	22	24	27
Réplicas	1	2	9	6	15	14	12	3	4	7	18	8	19	21	5	24	10	13	17	23	0	0	22	16	11	20	25
Tempo de Resposta	21	23	11	20	4	12	7	25	24	13	3	18	6	10	17	2	19	8	14	9	0	0	1	16	22	15	5
Menções	2	3	8	14	4	1	10	6	7	12	13	11	17	9	16	15	18	21	22	5	27	25	19	20	24	23	26
Seguidores	1	7	13	3	11	10	8	2	8	19	4	16	18	5	21	17	20	12	9	14	26	27	24	15	22	23	25
Pontuação	30	39	42	52	55	58	63	63	67	68	68	71	84	89	89	93	108	108	111	113	121	122	130	135	149	152	154

Figura 1: Gráfico das pontuações das métricas analisadas

Além das métricas, foi calculada a polaridade média entre todas as cidades utilizadas para construir o mapa no dashboard *web* (Fig. 2 e Fig. 3). Neste dashboard, pode-se visualizar a polaridade dos sentimentos dos *tweets* que mencionam os perfis do *Twitter* em análise. Além disso, observa-se que as cidades com melhor média de polaridade positiva foram melhores classificadas quanto *e-Participação*. Isto demonstra que quanto maior a interação entre a prefeitura com os cidadãos, melhor será o sentimento positivo da população em relação aos seus gestores.

Figura 2. Ilustração das polaridades de sentimentos processadas pelo Spark (22/05/2016)

Figura 3. Ilustração das polaridades de sentimentos processadas pelo Storm (23/05/2016)

VII. DISCUSSÃO

No sentido de avaliar a plataforma proposta, alguns aspectos da instanciação da plataforma em relação as duas principais plataformas de processamento em tempo real disponíveis (Apache Spark and Apache Storm) foram analisadas.

Quanto ao requisito de Processamento de Grandes Volumes de Dados em Tempo Real, de acordo com a referência [24], o *Apache Spark* pode processar até 670k registros por segundo e por nó, enquanto que o *Storm* 155k.

No entanto, elas se diferenciam no requisito de processamento em tempo real. O *Spark* tem latência de 0.52s [26], 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 [25, 26], sendo por isso um sistema com processamento em tempo real. 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* [25], 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*.

Em relação a Tolerância a Falhas, ela é suportada no *Spark Streaming* via *Discretized Streams* (*DStreams*), que é um modelo de programação para processamento de *streams* distribuídas, através do método *parallel recovery* [18]. No qual, implementa-se o conceito *lineage*, permitindo as informações serem recuperadas paralelamente [25].

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 [25]. 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 [25].

Todo o processo descrito até acima 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 [16]. 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 [16].

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 [14]. Caso o nó esteja indisponível, ou, falhando continuadamente, suas *tasks* são então atribuídas a outro disponível no *cluster*, via *Nimbus*.

O *Nimbus* e os *supervisors*, armazenam seus estados no *Zookeeper*, podendo ser reinicializados sem perdê-los em caso de falha, se houver falha no *Zookeeper*, outra instância pode ser "eleita" para o lugar dele [14]. Se algum supervisor falhar, seus *workers* são reatribuídos pelo *Nimbus* a outro supervisor, mas, ficando impedido de receberem novas *tuplas* [12]. Por sua vez, o *Nimbus*, 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 ao *cluster* [14], assim como reatribuições [12].

O modelo de recuperação de falhas do *Spark* pode ser uma boa escolha caso a aplicação permita perda de dados, em prol de eficiência [18]. Caso contrário, é necessário configurar uma fonte externa para replicação dos dados que estão sendo processados. 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 processamento 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 analise de polaridade dos sentimentos, processando *stream* de *tweets*, é mais interessante o suporte a Tolerância a Falhas do *Storm*, principalmente devido longo tempo de execução dela.

Sobre a Garantia de processamento, 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 [16].

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 evento foi processado ou não, através de seus *IDs* e mensagens informando "*ack*", podendo haver duplicatas; o oposto ocorre na segunda alternativa, em que perdas são aceitas, processando os eventos no máximo uma vez [17]. A proposta do *Spark Streaming*, é mais interessante para o processamento de *stream* de *tweets*, por garantir que a informação será processada uma única vez, através do modelo *Exactly Once*.

No quesito Escalabilidade, ambos suportam clusterização, sendo o maior *cluster Spark* conhecido composto por 8k nós [16]. Embora não tenham sido encontradas fontes confiáveis divulgando informações sobre o maior *cluster Storm* existente, ele é capaz de processar um milhão de mensagens de 100 *byte*, por segundo e por nó [17].

Por fim, a respeito do Modelo de programação, no *Spark* os RDDs são conjuntos de dados que podem ser manipulados de forma distribuída, sendo realizadas 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* foram mais difíceis de abstrair na plataforma proposta quando comparados ao do *Storm*. O *Storm* propõe o conceito de Topologia, composta por *bolts* e spouts. Os *spouts* são responsáveis pela entrada dos *streams* da aplicação, processados posteriormente pelos *bolts*. Devido a essas abstrações, consequentemente, pode haver maior facilidade em programar utilizando o *Storm* do que o *Spark*.

VIII. TRABALHOS RELACIONADOS

As redes sociais são amplamente utilizadas para coletar informações sobre pessoas e eventos. Por exemplo, [27] e [28] descrevem abordagens que capturam tweets em um determinado raio à partir de um ponto de aplicação. Na sequencia são utilizados modelos probabilísticos para identificar problemas relativos a transito e sobre outros eventos. Nossa abordagem se baseia nas soluções apresentadas em [27] e [28] apresentando como contribuição adicional uma plataforma que suporta o uso de duas plataformas subjacentes para processamento em tempo real (Apache Spark e Apache Storm) para coleta de métricas de *e-Participação*.

Em termos de estudos sobre *e-Participação*, em [3], é desenvolvida uma análise das Cidades Inteligentes da Espanha, verificando se nelas há esforços para assegurar que os cidadãos tenham acesso as informações da cidade, podendo assim participar das problemáticas abordadas por seus respectivos governos. Visando alcançar esse objetivo, primeiramente foram analisadas, descritivamente, as principais Redes Sociais utilizadas nessas cidades, tendo como resultado *Facebook* e *Twitter*. Na sequência, foram definidas variáveis independentes (métricas das Redes Sociais), dependentes (*Social Network Ratio*) e de controle (tempo de existência do perfil da Rede Social e PIB), realizando sob elas uma análise exploratória, através de múltipla regressão linear.

Por fim, o estudo exploratório foi utilizado para determinar se as Cidades Inteligentes da Espanha faziam ou não uso das Redes Sociais, focando o nível de interação entre os cidadãos e seus governos locais. Como resultado dessa pesquisa, concluiu-se que, embora essas cidades sejam classificadas como inteligentes, é preciso haver maior *e-Participação* nas Redes Sociais, pois, esse é um dos principais meios pelos quais as pessoas têm se comunicado.

Outro trabalho com a mesma temática, citado em [2] realizou uma análise dos principais governos locais da UE, no quesito de *e-Participação* e dos fatores que a influenciam. Com essa proposta, foram analisadas 75 cidades, representando 85% da população Europeia, sendo também levado em consideração o estilo de gestão das autoridades.

A metodologia de estudo analisou, exploratoriamente, as Redes Sociais *Twitter*, *Facebook*, *LinkedIn, YouTube* e *Blogger*, construindo um Índice de Sofisticação não Exaustivo, composto por variáveis binárias. O índice contém os respectivos indicadores de cada Rede Social. Em seguida, foi realizada uma análise de regressão para identificar como o estilo de gestão do governo e a facilidade com que a sociedade adota novas tecnologias influenciam no índice elaborado.

Os resultados da pesquisa mostram que algumas das cidades analisadas, estavam com iniciativas de *e-Participação* muito imaturas, e que metade dos governos não tinham representação nas Redes Sociais. Sendo, por outro lado, massiva a presença de cidadãos nas Redes Sociais discutindo sobre assuntos relacionados a cidade, havendo um isolamento de partes que deveriam interagir entre si. Por fim, concluiu-se também que os estilos de gestão dos governos locais não influenciam na adoção ou não das Redes Sociais, mas sim a facilidade com que a sociedade adota novas tecnologias.

Diante disso, o presente trabalho se distingue dos trabalhos descritos em [2] e [3] por apresentar uma plataforma para processamento de grande volume de dados que esta disponível em [20] para coleta de métricas relacionadas a e-Participação. Além disso, foi feito um estudo comparativo entre a instanciação da plataforma proposta usando o *Apache Storm* e *Spark* que observou vantagens e desvantagens para cada abordagem em relação a vários aspectos.

IX. CONCLUSÕES

Esse artigo apresentou uma plataforma para processamento de grande volume de dados, capaz de obter métricas relacionadas ao nível de *e-Participação* das capitais das cidades brasileiras, que embora importantes, têm sido desconsideradas dos indicadores que classificam as cidades como inteligentes. Apesar disso, a transparência do nível de *e-Participação*, pode contribuir para que cidadãos e gestores trabalhem em direção ao desenvolvimento de Cidades Inteligentes.

Além disso, conclui-se que as prefeituras mais afastadas das grandes metrópoles, de acordo com as métricas coletadas, têm feito pouco uso de seus perfis no *Twitter*, indicando um

baixo nível de interação entre governos locais e cidadãos. Em contraste, as prefeituras com melhores classificações de *e-Participação*, encontram-se próximas as regiões litorâneas, com maior desenvolvimento econômico.

Nos aspectos do desenvolvimento das aplicações, observou-se que a ferramenta *Spark* é a mais indicada para processamento de *tweets* em *batch*, sendo o Storm mais aconselhado para o processamento de *stream* de *tweets*, devido a sua arquitetura de Tolerância a Falhas.

REFERÊNCIAS

[1] CLARKE, R. Smart Cities and the Internet of Everything: The Foundation for Delivering Next Generation Citizen Services. 2013.

[2] BONSÓN, E. et al. Local e-government 2.0: Social media and corporate transparency in municipalities. Government Information Quarterly, v. 29, n. 2, 2012.

[3] SAÉZ-MARTIN, A.; HARO-DE-ROSARIO, A.; CABA-PEREZ, C. A vision of social media in the Spanish smartest cities. Trans-forming Government: People, Process and Policy, v. 8, n. 4, 2014.

[4] EMC. The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things. 2014.

[5] Connected Smartcities. Ranking Connected Smartcities. 2015.

[6] KILLELEA, P. Web Performance Tuning. 2st. ed. [S.l.]: O'Reilly Media, 2002.

[7] MORAIS, T. Survey on Frameworks for Distributed Computing: Hadoop, Spark and Storm, 2015.

[8] COULOURIS, G. et al. Sistemas Distribuídos Conceitos e Projetos. 5st. ed. [S.l.]: Bookman, 2013.

[9] SOMMERVILLE, I. Engenharia de Software. 9th. ed. [S.l.]: Pearson, 2011.

[10] MAGALHÃES, L. Instâncias e mecanismos de participação em ambientes virtuais: análise das experiências de participação política online em políticas públicas. 2015. 39o Encontro Anual da Asso-ciação Nacional de Pós-Graduação e Pesquisa em Ciências Sociais.

[11] MACIEL, C.; ROQUE, L.; GARCIA, A. Democratic Citizen-ship Community: a social network to promote e-Deliberative process. 2009.

[12] HART, B.; BHATNAGAR, K. Building Python Real-Time Applications with Storm. 1st. ed. [S.l.]: Packt Publishing, 2015.

[13] SHIN, K. G.; RAMANATHAN, P. Real-time computing: A new discipline of computer science and engineering. Proceedings of the IEEE, v. 82, n. 1, 1994.

[14] JAIN, A.; NALYA, A. Learning Storm. 1st. ed. [S.l.]: Packt Publishing, 2014.

[15] NARSUDE, C. Real-Time Event Stream Processing? What are your choices? 2015.

[16] Apache Software Foundation. Apache Spark. 2016.

[17] Apache Software Foundation. Apache Storm. 2016.

[18] ZAHARIA, M. et al. Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters. 2012.

[19] MORAIS, T. Survey on Frameworks for Distributed Computing: Hadoop, Spark and Storm. 2015.

[20] DIAS, F. Repositório no Git Hub. 2016. Disponível em: <https://github.com/fcas>. Acesso em Maio 21, 2016.

[21] Apache Software Foundation. Apache OpenNLP. 2016.

[22] ESKOVEC, J.; RAJARAMAN, A.; ULLMAN, J. Mining of Massive Datasets. 2016.

[23] SILVA, M. et al. SentiLex-PT 01. 2016. [24] DAS, T. Large-scale near-real-time stream processing. 2016.

[24] ZAHARIA, M. et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. 2012.

[25] ZAHARIA, M. et al. Discretized Streams: A Fault Tolerant Model for Scalable Stream Processing. 2012.

[26] KOMNINOS, N. and SCHAFFERS, H. Smart Cities and the Future Internet in Europe, *Journal of the Knowledge Economy*, Vol.3, No 3, 2012.

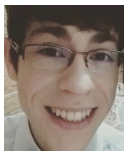
[27] DORAN, D., GOKHALE, S., and DAGNINO, A. Human sensing for smart cities. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '13)*. ACM, New York, NY, USA, 1323-1330, 2013.

[28] ANANTHARAM, P. et al. Extracting city events from social streams. *ACM Transactions on Intelligent Systems & Technology*, 2015.

[29] HANCKE, G. P., CARVALHO, S, B. and HANCKE Jr., G. P. 2013. The Role of Advanced Sensing in Smart Cities, Sensors, vol. 13, pp. 393–425.

[30] LOPEZ-HIGUERA, J. et al. Fiber Optic Sensors in Structural Health Monitoring. Journal of Lightwave Technology, vol. 29, no. 4, pp. 587–608, 2011.

[31] PERCHET, A. C. Reduce traffic congestion and improve mobility information services in urban places. [Online]. Available: http://bit.ly/XtEeqm



Felipe Cordeiro Alves Dias recebeu o título de bacharel em Engenharia de Software em 2016, pela Universidade Federal do Rio Grande do Norte (UFRN), com período sanduíche na Budapest University of Technology and Economics (Hungria), em Software Engineering, pelo programa federal Ciências sem Fronteiras. Atualmente, é aluno do programa de pós-graduação em Sistemas de Informação, da Universidade de São Paulo (USP). Seus interesses de pesquisa incluem Engenharia de Software, Cidades Inteligentes e Processamento e Análise de grande volume de dados.

Nélio Alessandro Azevedo Cacho recebeu o título de doutor em Ciência da Computação pela University of Lancaster (Inglaterra), Mestre e Bacharel em Ciência da Computação pela Universidade Federal do Rio Grande do Norte (UFRN). Atualmente é chefe do Departamento de Informática da Universidade Federal do Rio Grande do Norte e trabalha nas áreas de Engenharia de Software e Sistemas Distribuídos, com ênfase nos seguintes temas: Computação em Nuvem, Mecanismos de Tolerância a Falhas, Computação Ubíqua e Cidades Inteligentes.