*[[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 [1], fenômeno conhecido também como Tragédia dos Comuns [2], 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 [3].

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 [1].

Com essa quantidade de pessoas e dispositivos conectados, cerca de 44 trilhões de gigabytes serão gerados [4], os quais quando processados por sistemas inteligentes, ajudarão no surgimento de serviços de grande impacto no cotidiano de uma cidade [1]. Tal volume de dados, dobrará a cada dois anos até 2020 [4] principalmente devido ao crescimento do uso da Internet, smartphones, Redes Sociais, etc [1].

Cenário esse que demanda processamento de grande volume de dados, problemática na qual o conteúdo desse artigo se insere, propondo 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 [5]. 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 quanto ao 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 quantidade de dados existentes, conforme já mencionado anteriormente, 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 [6].

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) [7].

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 ocorra, por exemplo, por meio de replicação [8]. 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) [9].

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 [9]. 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 [3]. 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 [11].

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 [12]), 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.

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 [10]. 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 [10].

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 [10]. Além disso, a e-Participação não pode ser avaliada somente por seus aspectos técnicos, mas também quanto a capacidade de incrementar a democracia [10].

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 [10]. Além disso, a e-Participação é dividida em alguns níveis.

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 [10].

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*), 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*) 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*) [3].

# 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 [13, 14].

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) [13, 14].

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 [13].

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 [13]. 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) [13, 14].

Além dessa caracterização, 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 [15].

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 [15]. 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

Como base em [3], foram escolhidas as seguintes métricas, respectivas ao Twitter, dos perfis listados na Tab. I: média do número de tweets, seguidores, retweets (compartilhamento de um determinado tweet), 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. A aplicação [20] para o processamento de tweets e coleta dessas métricas foi desenvolvida na linguagem de programação Java, utilizando o framework para Web Services, Apache CXF [16], para expor seus serviços através de uma REST API, acessando a API do Twitter via a biblioteca Twitter4J [17].

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 [18].

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.

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.

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 explicadas **em**. Os valores dessa tabela são utilizados para criar o mapa contido na aplicação web.

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

No início da execução dessa aplicação [20], 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.

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 caixa baixa, as menções, *retweets*, são identificadas e removidas, assim como as referências a endereços de sites e símbolos. Em seguida a essa mineração inicial do texto, inicia-se o Processamento de Linguagem Natural.

Com esse proposito, usou-se a biblioteca OpenNLP [21] 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 e expressões normalmente abreviadas 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 [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 (se houver). Também, considera-se a presença de advérbios de negação, os quais modificam o significado do verbo, adjetivo e de outros advérbios [12], 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 do *tweet*, seu *id* e suas respectivas menções são armazenadas no banco de dados não relacional MongoDB. As polaridades são exibidas no mapa da aplicação web.

vI. Aplicação para processamento de stream de tweets, utilizando STORM

Nessa aplicação [20] é construída uma topologia, ilustrada na Fig. 3, composta por um Spout responsável pela conexão ao Twitter e coleta dos tweets, tendo como filtro o nome das contas das prefeituras no Twitter. 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.

vII. Revisão da literatura

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 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 [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 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 [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, há suporte no Spark Streaming via Discretized Streams (DStreams), que é um modelo de programação para processamento de streams distribuídas, através do método parallel recovery [27]. 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 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 [28]. 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 [28].

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 [30]. 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 [30]. Se algum supervisor falhar, seus workers são reatribuídos pelo Nimbus a outro supervisor, no entanto, ficando impedido de receberem novas tuplas [31]. 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 ao cluster [30], assim como reatribuições [31].

O modelo de recuperação de falhas do Spark pode ser uma boa escolha caso a aplicação permita perda de dados, em prol de eficiência [27]. 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 será́ processado exatamente uma vez (*Exactly Once*), sem perdas, ou, duplicadas, via *parallel recovery*, levando em consideração as observações já́ mencionadas [28].

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 [29]. A proposta do Spark Streaming, nesse aspecto, 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, portando são escaláveis horizontalmente, sendo o maior cluster Spark conhecido composto por 8.000 nós [28]. 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 [29].

Por fim, a respeito do Modelo de programação, 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 Topologia, 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.

vII. Trabalhos relacionados

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 Produto Interno Bruto), 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, posto que, atualmente, esse é um dos principais meios pelos quais as pessoas tem se comunicado.

Outro trabalho com a mesma temática, citado em [32] realizou uma análise dos principais governos locais da União Europeia, no quesito de e-Participação e dos fatores que a influencia. Com essa proposta, foram analisadas 75 cidades, representando 85% da população Europeia, sendo também levado em consideração o estilo de gestão de suas respectivas autoridades.

A metodologia de estudo analisou, exploratoriamente, as Redes Sociais Twitter, Facebook, LinkedIn, YouTube e blogs da Google (Blogger), construindo um Índice de Sofisticação não Exaustivo, composto por variáveis binárias. A construção desse 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 (na época do estudo), estavam com iniciativas de e-Participação muito imaturas, além disso, que metade dos governos não tinham representação nas Redes Sociais. Sendo, por outro lado, massiva a presença de cidadãos nas Redes Sociais discutindo sobre a politica local, entrega de serviços, e outros 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.

Saindo do contexto de e-Participação, a referência [19] realiza uma comparação entre o Apache Storm e Spark, relacionando-se com o estudo comparativo dessa monografia. Nela, o autor cita os casos de uso de cada uma delas, suas respectivas arquiteturas, comparando-as quanto ao modelo de processamento, latência, tolerância a falhas (especificamente, garantia de processamento), integração com processamento em batch e a diferentes linguagens.

vII. resultados

O gráfico exibido na Fig. 1 resume as métricas das tabelas contidas em [20], nele a cada métrica é atribuída uma pontuação, equivalente a classificação contida em [20], ou seja, o 27o lugar representa 27 pontos; consequentemente, quanto menos 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 pelo Connected Smart Cities, que classificou algumas das cidades brasileiras como Cidades Inteligentes[5] 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 as 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 |

1As cidades ausentes não estão classificadas como inteligentes ou não são capitais.

2Ranking geral composto por 50 cidades.

3Colocação como Cidade Inteligente.

4Colocação em e-Participação. O sinal de subtração indica desclassificação por ausência de uma das métricas analisadas.

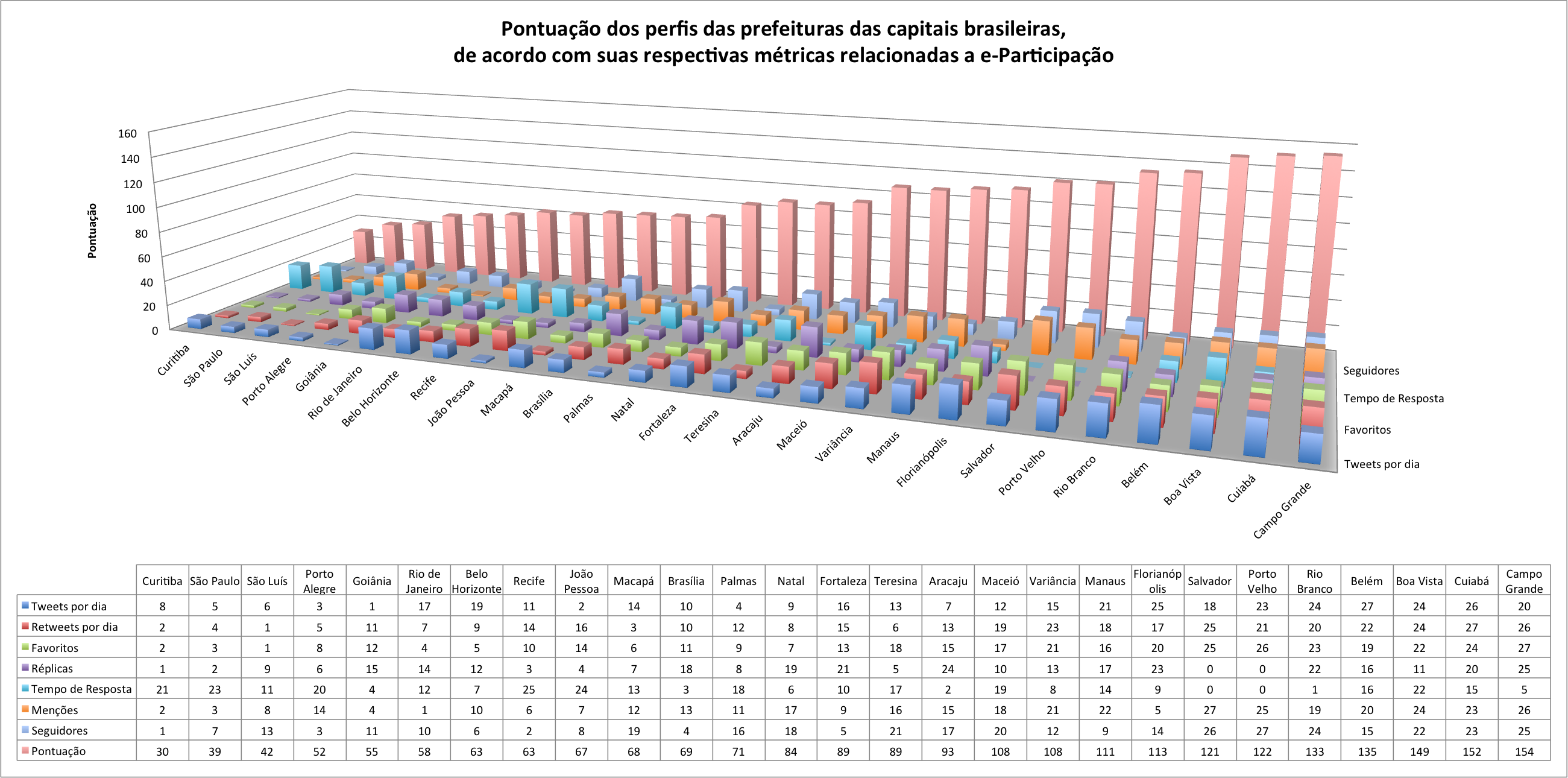


Figura 1: Gráfico das pontuações das métricas analisadas



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)

Com base nos resultados obtidos, para cada polaridade foi calculada a media entre todas as cidades, utilizada para construir o mapa da aplicação web (Fig. 2 e Fig. 3). No qual, pode-se visualizar a polaridade dos sentimentos dos tweets que mencionam os perfis do Twitter em análise. Além disso, observa-se que as cidades melhores classificadas quanto e-Participação interagiram mais com suas respectivas prefeituras, reforçando os primeiros resultados encontrados.

Quanto ao estudo comparativo entre o Storm e o Spark, podemos dizer que a primeira opção, respectivamente, é melhor de ser utilizada como aplicação para processamento de streams de tweets. Apesar do valor de throughput do Spark ser superior, e dele fornecer um modelo de processamento de dados mais eficiente, a arquitetura de Tolerância a Falhas do Storm precisa ser priorizada, devido ao longo tempo de execução da aplicação.

viii. conclusões

Esse artigo apresentou uma aplicação 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 ajam em direção ao desenvolvimento de Cidade Inteligentes.

Além disso, conclui-se que muitas prefeituras do Brasil, principalmente as afastadas das grandes metrópoles, de acordo com as métricas coletadas, tem feito pouco uso de seus perfis no Twitter, indicando um baixo nível de interação entre seus respectivos 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 melhor indicada para processamento de tweets em *batch*, enquanto que o Storm é mais aconselhado para o processamento de stream de *tweets*, principalmente 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 CitizenServices. 2013. Disponível em: <http://www.cisco.com>. Acesso em Maio 07, 2016.

[2] HARDIN, G. The tragedy of the commons. Science, v. 162, 1968. Disponível em: <http://science.sciencemag.org>. Acesso em Maio 08, 2016.

[3] SAÉZ-MARTÍN, A.; ROSSARIO, A. Haro-de; CABA-PEREZ, C. A vision of social media in the spanish smartest cities. Transforming Government: People, Process and Policy, v. 8, n. 4, 2014.

[4] SAÉZ-MARTÍN, A.; ROSSARIO, A. Haro-de; CABA-PEREZ, C. A vision of social media in the spanish smartest cities. Transforming Government: People, Process and Policy, v. 8, n. 4, 2014.

[5] Connected Smartcities. Ranking Connected Smartcities. 2015. Disponível em: <http://www.connectedsmartcities.com.br>. Acesso em Maio 08, 2016.

[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. Disponível em: <https://www.emc.com>. Acesso em Maio 11, 2016.

[8] COULOURIS, G. et al. Sistemas Distribuídos Conceitos e Projetos. 5st. ed. [S.l.]: Bookman, 2013. [9] SOMMERVILLE, 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 Associação Nacional de Pós-Graduação e Pesquisa em Ciências Sociais.

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

[12] PRIBERAM. Definição de deliberação. 2016. Disponível em: <https://www.priberam. pt>. Acesso em Maio 14, 2016. [13] SHIN, K. G.; RAMANATHAN, P. Real-time computing: A new discipline of computer science and egineering. Procedings og the IEEE, v. 82, n. 1, 1994.

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

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

Disponível em: <https://www.datatorrent.com>. Acesso em Maio 5, 2016.

[16] Apache Software Foundation. Apache CXF. 2016. Disponível em: <http://cxf.apache. org>. Acesso em Maio 20, 2016.

[17] TWITTER4J. 2016. Disponível em: <http://twitter4j.org/>. Acesso em Maio 20, 2016.

[18] MongoDB. 2016. Disponível em: <https://www.mongodb.com>. Acesso em Maio 20,

2016.

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

Storm. 2015. Disponível em: <https://www.emc.com>. Acesso em Maio 11, 2016.

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

[22] LESKOVEC; RAJARAMAN; ULLMAN, 2016

[23] SILVA, M. et al. SentiLex-PT 01. 2016. Disponível em: <http://dmir.inesc-id.pt/ project/SentiLex-PT\_01>. Acesso em Maio 21, 2016. [24] DAS, T. Large-scale near-real-time stream processing. 2016. Disponível em: <https://goo.gl/QSQ5uK>. Acesso em Maio 22, 2016. [25] ZAHARIA, M. et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction forIn-Memory Cluster Computing. 2012. Disponível em: <http://www-bcf.usc.edu/ ~minlanyu/teach/csci599-fall12/papers>. Acesso em Maio 23, 2016.

[26] ZAHARIA, M. et al. Discretized Streams: A Fault Tolerant Model for Scalable Stream Processing. 2012. Disponível em: <http://www.eecs.berkeley.edu/Pubs/TechRpts/ 2012>. Acesso em Maio 22, 2016.

[27] ZAHARIA, M. et al. Discretized Streams: An Efficient and Fault-Tolerant Model forStream Processing on Large Clusters. 2012. Disponível em: <http://www.eecs. berkeley.edu/~haoyuan/papers>. Acesso em Maio 23, 2016. [28] Apache Software Foundation. Apache Spark. 2016. Disponível em: <http://spark. apache.org>. Acesso em Abril 22, 2016.

[29] Apache Software Foundation. Apache Storm. 2016. Disponível em: <http: //storm.apache.org>. Acesso em Abril 30, 2016.

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

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

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

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)