Arquiteturas de Dados para Cidades Inteligentes baseadas em Internet das Coisas

Gabriela Biondi, Fabrizio Borelli, Carlos Kamienski, Ronaldo Prati
Universidade Federal do ABC (UFABC)
[gabriela.biondi, fabrizio.borelli,carlos.kamienski, Ronaldo.prati]@ufabc.edu.br

Resumo: As Cidades Inteligentes visam aplicar recursos de IoT para melhorar a qualidade de vida de uma comunidade como um todo. O armazenamento e processamento correto do grande volume de dados gerado são essenciais para combater de forma eficiente os principais problemas que uma cidade enfrenta. O objetivo deste artigo é propor uma abordagem de modelagem de dados relacional que garante três características que a modelagem de dados de uma cidade inteligente precisa apresentar: ser (1) genérica, (2) rastreável e (3) escalável, e fazer a avaliação de desempenho deste modelo de dados dentro de uma arquitetura de software.

1. Introdução

As Cidades Inteligentes (do inglês Smart Cities) visa aplicar os recursos de IoT não apenas para melhorar a qualidade de vida de um indivíduo, mas sim de uma comunidade como um todo, criando soluções que promovam cidadania, sustentabilidade, integração social, inclusão e participação, através do uso de tecnologias inovadoras [1].

Um dos obstáculos encontrados nesta área de pesquisa é o processamento e análise do grande volume de dados gerado por esses dispositivos. Além disso existe também o desafio de criar uma modelagem para armazenar esses dados, que seja genérica o suficiente para se adequar a diferentes cenários, rastreável para que se possa associar os acontecimentos e escalável, para não gerar gargalos na aplicação. As modelagens encontradas na literatura apresentam soluções específicas, pouco detalhadas, e quase sempre o conhecimento tácito e o trabalho manual de um cientista de dados são necessários.

O objetivo deste artigo é propor uma modelagem de dados para cidades inteligentes que seja genérica, rastreável e escalável, e fazer a avaliação de desempenho deste modelo de dados dentro de uma arquitetura de software.

2. Fundamentação Teórica

Internet of Things (IoT): É uma revolução tecnológica, que possibilita a conexão à Internet de todos os objetos do nosso cotidiano e que pretende fazer com que todos os nossos dispositivos atuais se transformem em objetos de computação ubíqua, melhorando, assim, a forma que nos relacionamos com nossos dispositivos, o jeito que nós os utilizamos e, até mesmo, a comunicação entre eles próprios [2].

Smart Cities: Um projeto de cidade inteligente se difere de um projeto de cidade tradicional por trazer conceitos de sustentabilidade, compartilhamento, inteligência e inovação através de diversas tecnologias, como IoT, por exemplo [3]. Neste artigo, exploraremos três cenários: iluminação pública; coleta de lixo; e transporte público.

Modelagem de Dados: Com o aumento do número de sensores que permeiam nossas vidas, a gestão eficiente da enorme quantidade de dados dos sensores está se tornando cada vez mais importante. Um dos desafios é o armazenamento desses dados em uma modelagem genérica, rastreável e escalável. Existem vários tipos de modelagens de dados, as mais usadas nos dias de hoje são as modelagens relacionais (SQL) e as modelagens não-relacionais (NoSQL). A modelagem relacional é a mais tradicional, foi proposta nos anos 70 por Edgar Frank Codd, e se baseia na ideia de agrupar os dados em tabelas que se relacionam entre si. Por esse motivo, esta será o primeiro tipo de modelagem testada. As modelagens NoSQL serão testadas nestes cenários em trabalhos futuros.

3. Materiais e Métodos

Para que o modelo seja genérico e se adeque a diferentes tipos de cenários, este trabalho de pesquisa propõem o uso de um conjunto de onze entidades genéricas que se relacionam entre si. São elas: Place, Subject, Resource, Service, Consumable, Action, Decision, Schedule, Fusion, Context e Result. Cada uma destas entidades foi pensada para representar elementos genéricos de diferentes cenários típicos de uma Cidade Inteligente.

O armazenamento dos dados dessas onze entidades é feito em um modelo relacional, que com a sua integridade, garantem a rastreabilidade dos eventos que ocorrerem na cidade. Na Figura 1 este modelo é apresentado. As onze entidades utilizadas no modelo genérico são representadas na cor verde, e armazenam os registros de cadastro de cada uma dessas entidades. As entidades representadas na cor vermelha armazenam registros de log processados. As entidades representadas na cor azul são entidades de tipagem ou associativas, criadas para garantir a integridade do modelo, permitindo assim a rastreabilidade dos eventos. As entidades representadas na cor amarela armazenam os registros de log dos sensores.

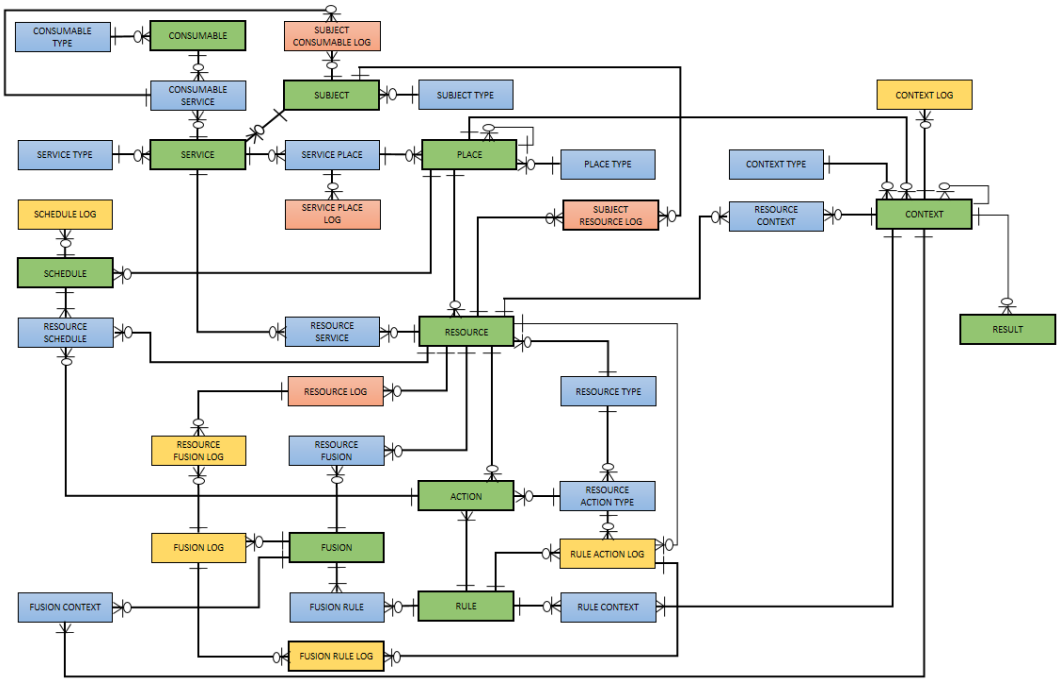

Figura 1: Modelo SQL Proposto

A escalabilidade foi mensurada através de uma avaliação de desempenho com duas arquiteturas de software de modo que, comparando os resultados das duas arquiteturas, podemos estimar o custo computacional do modelo de dados proposto neste artigo. Os elementos usados nestas arquiteturas são:

- Fusão (Fuser): responsável pela fusão de dados, que nada mais é do que a utilização de um conjunto de técnicas para combinar dados de múltiplas fontes ou estatísticas computacionais.
- Motor de Inferência (Reasoner): infere consequências lógicas de um conjunto de fatos.
- MQTT (Message Queuing Telemetry Transport): encapsula a comunicação com os recursos, interfaceando a comunicação com os sensores.
- Sensores: os sensores foram emulados usando o SenSE [4], um gerador de dados de IoT open-source.
- Atuadores: responsável por receber o resultado da inferência e atuar no ambiente.
- Buffer: armazena os dados temporariamente, enquanto o banco de dados estiver processando outras queries.
- DeterministicTracker: algoritmos de rastreamento fazem consultas complexas no banco de dados relacional para descobrir a ligação entre todos os acontecimentos na cidade.
- SQL: armazena dados da plataforma através de um sistema gerenciador de banco de dados relacional.

Na primeira arquitetura, ilustrada na Figura 2, os dados dos sensores são enviados via MQTT para o gerenciador de contexto, que fará as fusões, aplicará as regras e enviará as ações aos atuadores. Esta configuração da plataforma não garante a característica "genérica", pois não há armazenamento de dados. Sendo assim, cada novo sensor precisaria ser modelado manualmente para se integrar a plataforma.

A segunda arquitetura de testes, ilustrada na Figura 3, acrescenta o armazenamento de dados. O buffer usado no armazenamento SQL garante que não será criado um gargalo para fazer um armazenamento online, pois este processo pode ser feito em batch.

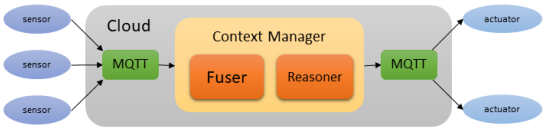

Figura 2: Arquitetura 1

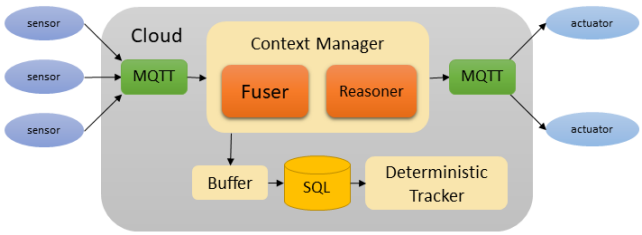

Figura 3: Arquitetura 2

Para executar os experimentos, o gerenciador de contexto foi implantado em um servidor dedicado com uma CPU Intel(R) Xeon(R) E3-1240 V2 @ 3,40GHz, 8 GB de memória RAM, executando o Ubuntu 14.04. A ferramenta SenSE foi executada em uma máquina virtual em outro servidor junto com o aplicativo de medição de de tempo. A configuração do segundo servidor é Intel Xeon E5520 2.27 GHz Octa Core(Virtual), com 11 GB de RAM.

Os testes de avaliação de desempenho usarão dados simulados para sensores nos três cenários descritos anteriormente: iluminação pública, transporte público e coleta de lixo público. Os sensores emulados pelo SenSE para estes cenários são de dois tipos: movidos por tempo e movidos por evento. Os sensores movidos por tempo enviam dados periodicamente para relatar o estado atual. Já os sensores movidos por eventos enviam dados apenas se o estado atual mudar.

- Iluminação Pública de Ruas: A implementação do cenário de iluminação pública de ruas considerou o uso de sensores de presença apenas para detectar a passagem de veículos nas vias. Estes sensores, movidos por evento, foram modelados com base nos números coletados na Av. Paulista – SP, pela CET (Companhia de Engenharia de Trafego) em maio de 2017. A Av. Paulista possui 2.700 metros de extensão e é equipada com 54 postes. O horário de pico de veículos, ou seja, o horário em que mais trafegam veículos pela avenida é das 9:00 às 10:00, com aproximadamente 6.884 veículos por hora.
- Transporte público: A implementação do cenário de transporte público do metrô, considerou o uso de sensores de presença apenas para detectar a passagem do metrô nas estações. Estes sensores, movidos por evento, foram modelados com base nos números coletados na linha azul dos metrôs de São Paulo – SP, pela Companhia do Metropolitano de São Paulo em 2017. A linha azul possui 23 estações distribuídas em 20,4 km de extensão. O horário de pico de passageiros, ou seja, o horário em que mais pessoas utilizam essa linha é das 8:00 às 9:00, quando a frequência entre os trens é de 1 minuto e 59 segundos. A linha amarela possui 11 estações distribuídas em 12,8 km de extensão. O horário de pico de passageiros, ou seja, o horário em que mais pessoas utilizam essa linha é das 8:00 às 9:00, quando a frequência entre os trens é de 2 minuto e 26 segundos. A linha verde possui 14 estações distribuídas em 14,7 km de extensão. O horário de pico de passageiros, ou seja, o horário em que mais pessoas utilizam essa linha é das 8:00 às 9:00, quando a frequência entre os trens é de 2 minuto e 25 segundos.
- Coleta de lixo público: A implementação do cenário de coleta de lixo em lixeiras de vias públicas, considerou o uso de sensores de presença apenas para detectar o nível de lixo dentro da lixeira. Estes sensores, movidos por tempo, foram modelados com base nos números lixeiras da Av. Paulista – SP. A Av. Paulista possui 2.700 metros de extensão e é equipada com 200 lixeiras. O horário de pico de pedestres, ou seja, o horário em que mais trafegam pessoas pela avenida é das 12:00 às 13:00, com aproximadamente 3.600 pedestres por hora. A leitura dos níveis das lixeiras é feita a cada 60 minutos.

4. Resultados e Discussões

Para realização dos experimentos todos os três cenários foram amplamente instanciados no banco de dados SQL, usando o SBDB (Sistema Gerenciador de Banco de Dados) PostgreSQL. Os dados instanciados foram gerados automaticamente pelo SenSE, mostrando qualitativamente que o modelo garante a características (1) ser genérico, pois conseguiu armazenar adequadamente os dados dos três cenários sem precisar de novas entidades.

Além disso, foram desenvolvidas 3 aplicações de rastreamento em Java (Tracker): o primeiro para rastrear o ambiente (Environment Tracker), o segundo para rastrear os consumíveis trocados (Consumable Tracker) e o terceiro para rastrear a agenda (Schedule Tracker). Após a sua execução sobre os dados instanciados inicialmente a partir do SenSE, 100% dos registros buscados foram encontrados, mostrando quantitativamente que o modelo garante a característica (2) ser rastreável.

Por fim, foram realizados experimentos de avaliações de desempenho para medir quantitativamente a característica (3) ser escalável. Esses experimentos medem o tempo de resposta entre uma ação (os dados dos sensores são enviados para a plataforma) e a reação do contexto (os dados são enviados da plataforma para os atuadores), considerando a taxa de entrega de mensagens no sistema.

O SenSE foi usado para simular vários números de sensores com diferentes tipos de envio de dados (sensores movidos por tempo e sensores movidos por evento), para entender o possível comportamento do gerenciador de contexto em um ambiente urbano inteligente. A ferramenta SenSE foi configurada para trabalhar baseada em dados reais, coletados na cidade de São Paulo.

Os sensores movidos por eventos do SenSE enviam mensagens de acordo com uma distribuição de Poisson e o parâmetro λ define a taxa de chegada de pessoas em um evento simulado. Observou-se que a taxa de chegada de pessoas em um evento típico (por exemplo, pessoas que chegam a um estádio) variam de acordo com três momentos: a) Início: pouco antes e um pouco após o início do evento as pessoas chegam em quantidade moderada, já que geralmente as pessoas tendem a chegar antecipadamente ou um pouco atrasadas; b) Meio: durante o evento a

taxa de chegada é baixa, já que apenas poucas pessoas chegam e saem no meio do evento; c) Fim: após o final do evento o fluxo de pessoas é alto, já que todas as pessoas saem ao mesmo tempo. Dentro nos experimentos, utilizamos esses três períodos diferentes para modificar a taxa de chegada de tal forma que se λ é a taxa no início do experimento, no meio é diminuído para λ /10 e no fim ele aumenta para 3λ.

Cada experimento foi replicado 30 vezes e o intervalo de confiança assintótica foi calculado com uma confiança nível de 99%. A Tabela 1 mostra as cargas de trabalho usadas no SenSE em cada tipo de sensor para todos os experimentos.

Tabela 1: Cargas dos Experimentos

	Iluminação Pública	Transporte Público	Lixo
Carga Baixa	50 postes simulados com taxa de chegada de 1.721 veículos/h	1 linha azul do metrô com intervalo de 476s entre os trens; 1 linha verde do metrô com intervalo de 580s entre os trens; 1 linha verde do metrô com intervalo de 584s entre os trens;	50 lixeiras
Carga Média	50 postes simulados com taxa de chegada de 3.442 veículos/h	1 linha azul do metrô com intervalo de 238s entre os trens; 1 linha verde do metrô com intervalo de 290s entre os trens; 1 linha verde do metrô com intervalo de 292s entre os trens;	100 lixeiras
Carga Alta	50 postes simulados com taxa de chegada de 6.884 veículos/h	1 linha azul do metrô com intervalo de 119s entre os trens; 1 linha verde do metrô com intervalo de 145s entre os trens; 1 linha verde do metrô com intervalo de 146s entre os trens;	200 lixeiras

Na Figura 4 é possível observar que a primeira arquitetura apresenta um comportamento linear, aumentando conforme a carga de trabalho aumenta. Na Figura 5 é possível observar que a segunda arquitetura também apresenta um comportamento linear, aumentando conforme a carga de trabalho aumenta. Porém, para fins comparativos, é possível observar que na segunda arquitetura, onde todos os registros são armazenados em banco de dados, o tempo de execução é praticamente o dobro do tempo de execução da primeira arquitetura. Em outras palavras, acrescentando a modelagem SQL a arquitetura está registrando todos os acontecimentos, ganhando a capacidade de rastrear eventos e oferecendo um modelo com entidades genéricas que se adapta a diferentes cenários típicos de uma cidade inteligente. Tudo isso sem necessariamente comprometer o desempenho da plataforma, pois o tempo de processamento das mensagens está abaixo de 19ms, mesmo com a carga de trabalho alta.

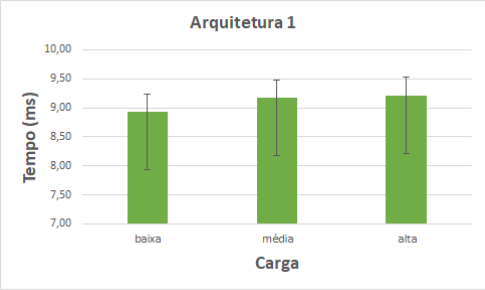

Figura 4: Resultados da Arquitetura 1

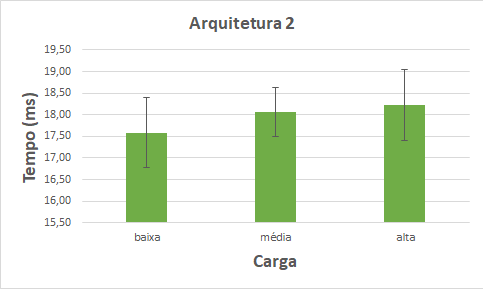

Figura 5: Resultados da Arquitetura 2

Vale ressaltar que o objetivo da avaliação é medir o impacto que o uso do modelo proposto gera, por isso não houve comparação com experimentos externos.

5. Conclusão

As principais contribuições deste artigo foram: (1) definição de três características que um modelo de uma cidade inteligente precisa apresentar; (2) construção de um modelo que apresente essas características; (3) definição de duas arquiteturas para comparar o impacto do modelo proposto; e (4) resultados da avaliação de desempenho.

A modelagem relacional garantiu a integridade dos registros, permitindo assim a rastreabilidade dos eventos. A definição de onze entidades garantiu que o modelo fosse genérico e se adequasse a diferentes cenários típicos de uma cidade inteligente. O desempenho apresentado em ambas as arquiteturas mostrou um comportamento linear conforme aumentamos a carga de trabalho, mostrando que o modelo possui uma escalabilidade adequada.

O modelo proposto já está sendo testado com sucesso em outros projetos e novos experimentos estão sendo feitos para avaliar o seu desempenho sob outros critérios, por exemplo usando uma modelagem NoSQL.

6. Referência

[1] Monroy-Hernández, A., Farnham, S., Kiciman, E., Counts, S., and Choudhury, M. D. (2016). Smart societies: From citizens as sensors to collective action. CoRR, abs/1605.08844.

[2] Weiser, M. (1991). The Computer for the 21st Century. Scientific American, 265, 94-104.

[3] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Context aware computing for the internet of things: A survey. IEEE Comm. Surveys Tuts, 16(1):414–454.

[4] Zyrianoff, I., Borelli, F., and Kamienski, C. A. (2017). SenSE (Sensor Simulation Environment): Uma ferramenta para geração de tráfego IoT em larga escala. SBRC 2017. Salão de Ferramentas.