Uma Contribuição à Ciência de Contexto no Middleware EXEHDA Explorando Fog Computing

Leonardo João[1], Verônica Tabim[2], Anderson Cardoso[2], Roger Machado[1], João Lopes[3], Gerson Cavalheiro[1], Ana Marilza Pernas[1], Adenauer Yamin[12]

[1]Universidade Federal de Pelotas (UFPel)

[2]Universidade Católica de Pelotas (UCPel)

[3]Instituto Federal Sul-rio-grandense (IFSul)

{ldrsjoao,rdsmachado,gerson.cavalheiro,marilza,adenauer}@inf.ufpel.edu.br
{veronica.tabim,anderson}@ucpel.edu.br
joaolopes@cavg.ifsul.edu.br

Abstract. *Advances in Internet of Things (IoT) have enabled interoperation between various intelligent devices, collecting and distributing contextual information, thus allowing more sophisticated models to provide context awareness. The use of middlewares is highlighted in the literature as an alternative to device management in distributed IoT infrastructure, as well as to make contextual processing more transparent to applications. Considering this scenario, the objective of this work is to present the Fog Computing proposal for the middleware EXEHDA, considering the use of computational edge devices. A case study was carried out in the area of Precision Viticulture (PV) evaluating the proposed architecture.*

Resumo. *Os avanços na Internet of Things (IoT) permitiram a interoperação entre diversos dispositivos inteligentes, coletando e distribuindo as informações contextuais, assim possibilitando modelos mais sofisticados para prover a ciência de contexto. O uso de middlewares é destacado na literatura como alternativa para gerenciamento de dispositivos na infraestrutura distribuída IoT, bem como para tornar o processamento contextual mais transparente para as aplicações. Considerando este cenário, o objetivo desse trabalho é apresentar a proposta de Fog Computing para o middleware EXEHDA, considerando o uso de dispositivos das bordas computacionais. Foi realizado um estudo de caso na área de Viticultura Precisão (VP) avaliando a arquitetura proposta.*

1. Introdução

Nos últimos anos, a rápida evolução nas áreas de dispositivos embarcados e redes de computadores, promoveu a disponibilidade de sistemas ciber-físicos com elevados níveis de conectividade. A sinergia decorrente da interoperabilidade de objetos inteligentes, formados pelos sistemas ciber-físicos, vem constituindo a *Internet of Things* (IoT) [Peralta et al. 2017].

As aplicações introduzidas pelo cenário IoT, embora onipresentes, são processadas em um ambiente computacional amplamente distribuído, e requerem comportamento autônomo, minimizando o envolvimento de seus usuários. Para atender este requisito estas aplicações necessitam ter ciência dos dados contextuais que lhe interessam e, quando

for o caso, reagirem aos mesmos. Esta classe de sistemas computacionais, reativos ao contexto, abre perspectivas ao desenvolvimento de aplicações mais ricas, elaboradas e complexas, e que exploram a natureza dinâmica das modernas infraestruturas computacionais [Botta et al. 2016].

No cenário da IoT a tecnologia de *Cloud Computing* vem oferecendo elevados níveis de disponibilidade e escalabilidade. No entanto, uma vez que a centralização é inerente à *Cloud Computing*, ela não é capaz de atender todos os requisitos do desenvolvimento de soluções na perspectiva da IoT, principalmente considerando infraestruturas distribuídas formadas por um elevado número de dispositivos embarcados [Aazam and Huh 2014].

Os dados coletados na infraestrutura computacional nem sempre precisam ser transmitidos para a nuvem. Eles podem ser processados localmente nos próprios dispositivos embarcados, por meio de políticas de filtragem e agregação, com o objetivo de economizar recursos de rede e armazenamento [Bonomi et al. 2014].

Visando melhorar a capacidade de agregação e abstração, os dispositivos embarcados podem requisitar dados de outros dispositivos em um mesmo espaço geográfico, colaborando para o processamento contextual e materializando assim as premissas de um novo paradigma da computação denominado *Fog Computing* [Privat et al. 2016].

Diferentes desafios estão presentes no provimento de suporte para as aplicações direcionadas a IoT considerando o paradigma de *Fog Computing*, dentre estes, temos o gerenciamento das informações coletadas através de dispositivos heterogêneos, e a interpretação destas informações considerando o contexto de interesse das aplicações [Bellavista et al. 2012]. As informações coletadas no sentido de promover a compreensão do contexto de interesse das aplicações, denominam-se neste trabalho de dados contextuais.

Neste sentido, este artigo tem como objetivo principal apresentar a abordagem projetada para fornecer suporte à Ciência de Contexto para o middleware EXEHDA [Lopes et al. 2014], explorando o paradigma da *Fog Computing*.

A principal contribuição desse trabalho é a concepção de uma arquitetura para o Subsistema de Reconhecimento de Contexto e Adaptação do EXEHDA, com a capacidade de lidar com a aquisição e processamento distribuído de dados contextuais na perspectiva da IoT, utilizando as bordas computacionais e assim viabilizando o emprego da abordagem de *Fog Computing*.

O texto do artigo está estruturado da seguinte forma. A seção 2 discute o escopo desse trabalho, sendo apresentado os temas relacionados a Ciência de Contexto, *Fog Computing* e *middleware* EXEHDA. Na sequência, a seção 3 descreve os trabalhos relacionados. A seção 4, apresenta a arquitetura desenvolvida. A seção 5, discute a avaliação da arquitetura, sendo apresentado os hardwares e softwares utilizados e ainda os resultados e discussões a respeito do estudo de caso. Finalmente, na seção 6 são discutidas algumas contribuições alcançadas ao final deste trabalho e os possíveis trabalhos futuros.

2. Escopo do Trabalho

A Ciência de Contexto refere-se a um modelo no qual o sistema computacional é capaz de verificar as características do meio no qual tem interesse e, quando necessário, reagir

as suas mudanças [Khattak et al. 2014]. No cenário da IoT, a Ciência de Contexto possui elevado significado, uma vez que existe a real necessidade do sistema tomar decisões autônomas nos mais diversos tipos de dispositivos embarcados que estão interoperando, considerando as especificações do usuário, ou aquelas inferidas pelo próprio sistema.

A *Cloud Computing* é uma das tecnologias mais promissoras para a materialização da IoT, por viabilizar sistemas escaláveis ao fornecer uma grande capacidade de armazenamento de dados e processamento [Vermesan and Friess 2014]. Apesar desses recursos, as soluções baseadas em nuvem exigem uma conexão ativa com a Internet tanto para o processamento de dados contextuais quanto para a atuação no ambiente. Essas duas operações exigem uma abordagem que garanta a confiabilidade operacional do sistema, mesmo no caso de desconexões [Stojmenovic and Wen 2014]. Deste cenário, decorre a motivação para o uso do *Fog Computing* no *middleware* EXEHDA.

A *Fog Computing* surgiu com a intenção de abordar desafios relacionados a baixa latência, ciência de localização e confiabilidade [Bonomi 2011]. A mesma é implementada nas bordas computacionais, normalmente com dispositivos embarcados. Essa infraestrutura, feita com os dispositivos mais próximos das necessidades do usuário, é considerada como uma nuvem mais próxima do usuário, proporcionando assim baixa latência, ciência de localização e possibilidade de aprimoramento da *Quality of Service* (QoS). Deste modo, a *Fog Computing* fornece armazenamento temporário, capacidade de raciocínio, auxiliando na tomada de decisão considerando os dados de contexto coletados.

O EXEHDA consiste de um *middleware* adaptativo ao contexto baseado em serviços, que visa criar e gerenciar um ambiente ubíquo, bem como promover a execução de aplicações sobre ele. O *middleware* vem sendo utilizado pelo *Laboratory of Ubiquitous and Parallel Systems* (LUPS) em frentes de pesquisa que abordam desafios da IoT [Souza et al. 2015, Davet et al. 2015].

O ambiente fornecido pelo EXEHDA é formado por equipamentos multi-institucionais, sendo composto por dispositivos de usuário e alguns equipamentos para suporte as demandas do próprio middleware, para tanto, cada dispositivo é instanciado considerando seu respectivo perfil de execução no middleware.

Na composição do ambiente podem estar presentes sistemas embarcados, os quais também são instanciados pelo seu respectivo perfil de execução do *middleware*. Conforme pode ser observado na Figura 1, o EXEHDA possui uma organização composta por um conjunto de células de execução. Estas células, no que diz respeito ao provimento de Ciência de Contexto, são compostas por Servidores de Contexto, Servidores de Borda e Gateways. O Servidor de Borda (SB) é responsável por interagir com o ambiente através de Gateways que gerenciam sensores e atuadores. Por sua vez, o Servidor de Contexto (SC) fornece as funcionalidades para a Ciência de Contexto. Esses servidores são alocados em células no ambiente gerenciado pelo EXEHDA.

Em cada célula podem existir vários Servidores de Borda e Gateways e um equipamento central (EXEHDAbase) onde é executado o Servidor de Contexto. A organização celular do EXEHDA tem por objetivo também assegurar a autonomia das instituições envolvidas.

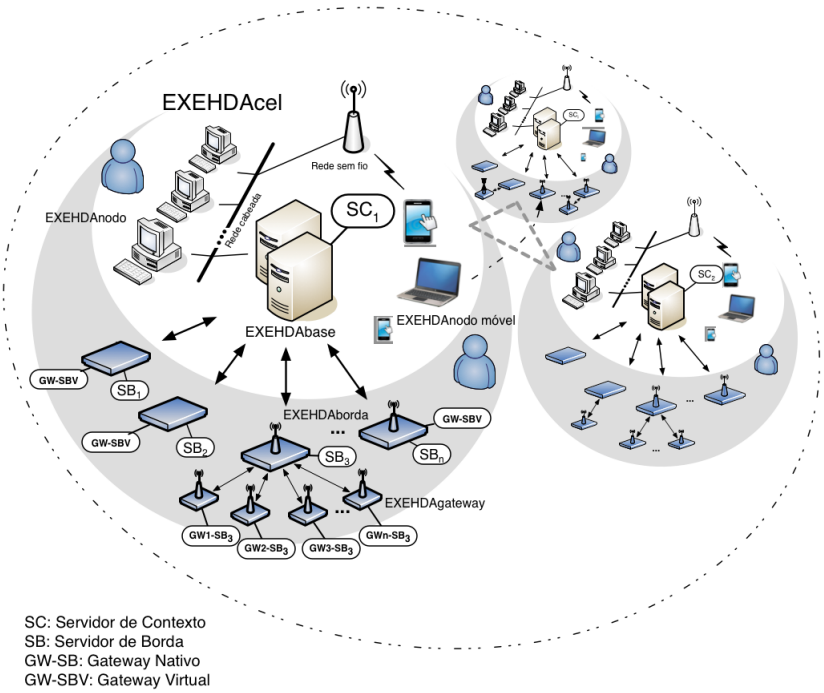

Figura 1. Ambiente Ubíquo gerenciado pelo EXEHDA

3. Trabalhos Relacionados

Nesta seção serão apresentados os trabalhos relacionados que foram identificados de acordo com os seguintes critérios: modernidade, desenvolvimento ativo e representatividade no cenário atual. Além dos critérios citados, levou-se em consideração requisitos relacionados ao tratamento de eventos em *Fog Computing*: distribuição; suporte a composição; escalabilidade; e gerência dinâmica.

WSO2 [Fremantle 2015] propõe uma série de soluções para a IoT, dentre elas uma arquitetura de referência com o objetivo de auxiliar desenvolvedores quanto a concepção de arquiteturas para o cenário da IoT. A arquitetura tem como objetivo fornecer um ponto de partida eficaz que contemple a maior parte dos requisitos de sistemas e projetos envolvendo IoT. No WSO2 são identificados alguns pontos necessários para soluções IoT, sendo classificados em cinco grupos: (i) conectividade e comunicação; (ii) gerenciamento de dispositivos; (iii) obtenção de dados e análise; (iv) escalabilidade; e (v) segurança. Além desses, pode ser mencionado ainda alta disponibilidade, predição e facilidade de integração.

Xively [Xively 2018] disponibiliza um *middleware* comercial, utilizando uma abordagem baseada em nuvem para tratar e armazenar os dados providos pelos dispositivos. O sistema é baseado no padrão arquitetural REST, disponibilizando os sensores e atuadores como recursos Web e interfaces padronizadas para a troca de dados através de uma estratégia de sensoriamento como serviço [Zaslavsky et al. 2013].

LinkSmart [LinkSmart 2018], anteriormente chamado Hydra, é uma plataforma de *middleware* baseada em *Service-Oriented Architecture* (SOA), desenvolvida na linguagem Java, que oferece suporte ao desenvolvimento de aplicações com base em

informações fornecidas por dispositivos físicos heterogêneos, disponibilizando interfaces de serviços Web para controle destes dispositivos.

Carriots [Carriots 2018] propõe uma plataforma para aplicações em IoT que utiliza serviços de nuvem para gerenciar dados providos por qualquer tipo de dispositivo, além de conectar dispositivos e outros sistemas, o que faz se alinhar ao conceito de *Plataformas as a Service*(PaaS) da computação em nuvem.

A sistematização das funcionalidades e dos perfis operacionais destes trabalhos relacionados, tem sido centrais nos esforços de pesquisa referentes as novas funcionalidades do *middleware* EXEHDA.

4. Arquitetura Desenvolvida

A proposta de *Fog Computing* para o *middleware* EXEHDA herda as características de Ciência de Contexto, empregando uma estratégia colaborativa entre a aplicação e o ambiente de execução, através da qual é oportuno ao programador individualizar políticas para gerenciar o comportamento de cada um dos componentes que constituem o software da aplicação. Na Figura 2 é apresentada a arquitetura desenvolvida para o Servidor de Borda do middleware EXEHDA, sendo mostrados os módulos que a compõem.

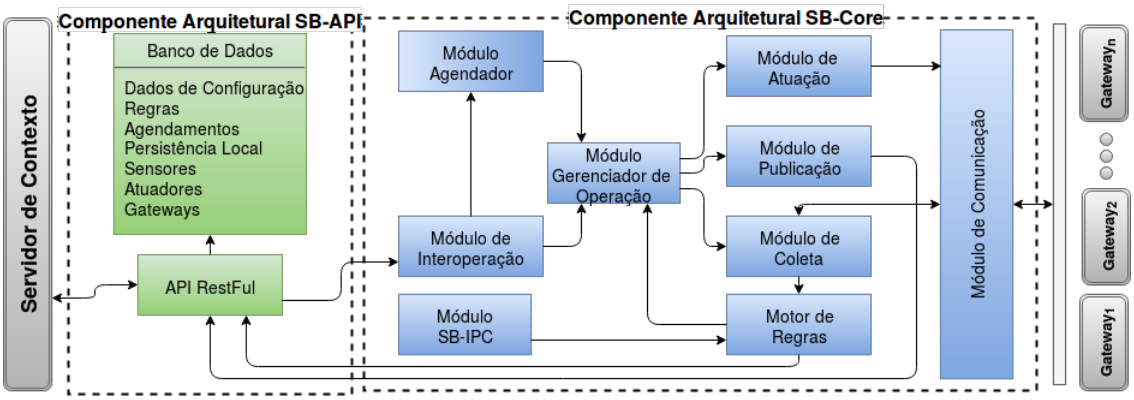

Figura 2. Módulos do Servidor de Borda.

4.1. Componente Arquitetural SB-API

A SB-API, implementada sobre um web-server, é responsável pelas requisições provenientes de clientes ativos no momento, criando novos eventos para as mesmas. Os dados são recebidos via HTTP por requisição REST pela SB-API, e enviados tanto para o banco de dados, como para o Módulo de Interoperação em formato JSON. Através de um servidor *RestFul* utilizando o *framework DJango*[1], as requisições são processadas nos padrões REST para o acesso às informações e aos serviços do Servidor de Borda.

Na Tabela 1 são apresentados os possíveis URI's (Uniform Resource Identifiers) de acesso à SB-API. Esses recursos podem ser acessados pelos seguintes endereços: *groups, users, manufacturers, gateways, actuators, baseParameters, contextServers, sensorsTypes, sensors, persistances, rules, schedules e topics*.

[1]https://www.djangoproject.com/

Tabela 1. Tabela URI do SBWebService

Verbo	URI	Descrição
GET	IP/	Lista todos os tipos de recursos
POST	IP/Recurso	Insere um recurso
PUT	IP/Recurso/ID	Atualiza informações do recurso
GET	IP/Recurso/ID	Informações do recurso
DELETE	IP/Recurso/ID	Deleta o recurso

4.2. Componente Arquitetural SB-Core

O SB-Core é responsável pela atuação e processamento dos dados cadastrados na SB-API, bem como pelo acionamento de regras sobre os eventos a serem tratados.

As mensagens transferidas entre cada um dos módulos estão formatadas em JSON, dessa forma ocorre a padronização na passagem dos argumentos necessários para atuação de cada etapa do Servidor de Borda. Quando a SB-API recebe dados do Servidor de Contexto, a mesma salva as informações no Banco de Dados, comunicando o SB-Core sobre uma possível inserção de um novo sensor ou de um agendamento. Os agendamentos são registrados para sensores previamente cadastrados. Na Tabela 2 podem ser visualizadas as URI's utilizadas.

Tabela 2. Tabela URI do SBCore

Verbo	URI	Descrição
POST	IP/sigSensor_add	Notifica o core sobre um novo sensor
POST	IP/sigSensor_delete	Notifica o core sobre a remoção de sensor
POST	IP/sigScheduler_add	Notifica o core sobre um novo agendamento
POST	IP/sigScheduler_delete	Notifica o core sobre a remoção de agendamento

Os módulos que constituem o componente arquitetural do SB-Core estão descritos a seguir.

Módulo Agendador

Gerencia os eventos periódicos cadastrados no banco de dados do SB-API, entre eles a leitura de dados dos sensores e controle dos atuadores. Por sua vez, os eventos aperiódicos produzidos por outros Servidores de Borda são recebidos através do Módulo de Interoperação. No controle das tarefas utilizou-se a biblioteca implementada em Python, *Advanced Python Scheduler*(APScheduler) [APScheduler 2018].

O *APScheduler* permite o cadastro de uma ação, considerando que a tal seja executada periodicamente ou uma única vez, permitindo a inserção e gerência de tarefas já cadastradas.

A biblioteca *APScheduler* possui três modos de gerência, entre eles: (i) **Cron-style scheduling**: é o modo mais completo, possui todos requisitos do *CRON* e funcionalidades de adicionar horário de início e de fim nas atuações de tarefas; (ii) **Interval-based execution**: executa as tarefas com intervalos regulares, com a opção de adicionar horário de início e término; e (iii) **One-off delayed execution**: executa uma única tarefa, determinada por data e horário específicos.

Módulo Gerenciador de Operações

Administra o fluxo de dados do agendador com os demais módulos, verificando o tipo de evento a ser realizado, entre eles: atuação, coleta e publicação. Após a verificação, transmite os dados necessários para processamento nos módulos seguintes.

Módulo de Atuação

Realiza a requisição de atuação através do Módulo de Comunicação, transmitindo uma mensagem JSON com os parâmetros necessários para identificar um determinado atuador conectado nos Gateways ativos.

Módulo de Coleta

Requisição de coleta ao Módulo de Comunicação, enviando uma mensagem no formato JSON com os parâmetros necessários para a ação de coleta. Após o recebimento dos dados, os mesmos são repassados ao Motor de Regras para o seu processamento dos dados contextuais.

Módulo de Publicação

Publica o dado contextual do ambiente no Servidor de Contexto e realiza o armazenamento temporário localmente. Entretanto, caso a publicação não seja efetivada por motivos de comunicação, os dados são armazenados localmente, e posteriormente quando a conexão for reestabelecida, os dados contextuais são transmitidos.

Motor de Regras

Processa os dados contextuais relacionando as informações com as regras cadastradas pelo usuário, tendo potencial de combinar o dado contextual de um sensor com outro e até mesmo com outro ambiente em monitoramento, ou mesmo uma requisição de atuação sob o ambiente. O Motor de Regras é implementado em Python, de código aberto e sendo possível alterá-lo para uso em fins específicos.

Módulo de Comunicação

Realiza as requisições de atuação e coleta do Servidor de Borda aos Gateways, disponibilizando uma única interface de comunicação com diversos protocolos existentes. Combinando o *UUID* do sensor ou atuador com o IP de acesso a este Gateway. Pode-se observar na Tabela 3, o formato das URI's utilizadas no acesso.

Módulo de Interoperação

Emprega um Servidor HTTP, recebendo notificações compostas de novos sensores e agendamentos cadastrados no Servidor de Contexto, repassando as informações ao Módulo Agendador em tempo real. Essas URI's de acesso, podem ser visualizadas na Tabela 2.

Módulo SB-IPC

Utiliza o protocolo de comunicação *Message Queue Telemetry Transport* (MQTT) para troca de dados. Este protocolo segue o modelo cliente/servidor, onde os dispositivos sensoriados são os clientes que se conectam a um servidor chamado Broker, usando TCP/IP. Os clientes podem subscrever em diversos tópicos, e são capazes de receber as mensagens

de diversos outros clientes que publicam neste tópico. Dessa forma, este módulo recebe dados de outros Servidores de Borda que são de seu interesse. Entre os dados recebidos, temos o dado contextual coletado e a regra relacionada a este dado. Os dados recebidos por este módulo são transmitidos ao Motor de Regras.

Tabela 3. Tabela URI do Gateway

Verbo	URI	Descrição
GET	IP/sensor?uuid=parametro	Coleta de dados do sensor físico
POST	IP/atuador?uuid=parametro	Ativa/desativa o atuador

5. Prototipação e Avaliação

A *Fog Computing* é implementada nas bordas computacionais do EXEHDA, transferindo parte do processamento para os sistemas embarcados localizados no ambiente do usuário final, sendo assim considerada uma nuvem computacional mais próxima da ocorrência dos eventos de interesse. Uma discussão detalhada da arquitetura do Servidor de Borda do EXEHDA pode ser encontrada em [Souza 2017].

A viticultura é uma frente de exploração agrícola que possui grande potencial econômico e social em diversas regiões do Brasil, e em particular no Rio Grande do Sul. A Viticultura de Precisão (VP) pode ser entendida como a gestão da variabilidade temporal e espacial das áreas cultivadas com o objetivo de melhorar o rendimento econômico da atividade agrícola [Braga and Pinto 2009].

Nesse sentido, uma das questões fundamentais na produção de vinhos de alta qualidade é o momento certo de irrigar. Para determinar o momento correto de irrigar é necessário avaliar as variáveis físicas do ambiente. No cenário em estudo, as variáveis contextuais consideradas são as seguintes: tensão de água no solo, temperatura e umidade do ar. Os comandos de atuação disparados através do processamento de regras, são os seguintes: alertas visuais e sonoros; envio de mensagens (SMS/e-mail); e atuação de transdutores elétricos para os acionamentos dos sistemas de irrigação.

Visando uma qualificação da informação e validação de consistência, foi considerado o uso de diversos sensores distribuídos ao longo das zonas de manejo. Cada zona de manejo é equipada com um Servidor de Borda capaz de gerenciar os Gateways e sensores atribuídos a estes, para que o controle de irrigação não seja inviabilizado por eventuais quedas de conexão de rede, ocasionando perda de contato com o Servidor de Contexto.

Através desse cenário pretende-se avaliar à capacidade de operação em *Fog Computing*. Atualmente, grande parte das áreas rurais dependem de conexões móveis através de rede de celular para conexão à Internet, muitas vezes sujeitas a conexões instáveis, baixa largura de banda e volume de dados mensais limitados através de franquias. Esses aspectos apontam para a necessidade de uso moderado do tráfego de dados através da Internet e o uso de estratégias que garantam a operação em momentos em que esse acesso não é possível.

5.1. Hardware e Software Utilizados

O Servidor de Borda foi concebido para ser disponibilizado nas instalações físicas através de um modelo Raspberry PI [2] B +, usando o sistema operacional Raspbian. Os Gateways são dispositivos de baixo poder computacional, utilizados como forma de comunicação entre os Sensores e Servidores de Borda. Visando o baixo consumo energético e o emprego de uma plataforma de software disseminada, para prototipação dos Gateways utilizou-se o processador ESP8266-12 [Kurniawan 2016], um dispositivo com suporte a comunicação WI-FI e programável através do framework de desenvolvimento da plataforma Arduino[3].

Para atender as demandas de sensoriamento utilizou-se a tecnologia 1-Wire. Esta tecnologia apresenta a transmissão de dados através de um único barramento, no qual todos os dispositivos são endereçáveis, apresentando robustez e flexibilidade no desenvolvimento de redes sensores cabeados.

O ambiente computacional concebido para avaliação deste estudo de caso é baseado no *Common Open Research Emulator* (CORE) [Ahrenholz et al. 2008]. Este framework *open-source* desenvolvido pela *Boeing Research and Development Division* (BR&T)[4] visa a emulação de ambientes computacionais, permitindo assim testes largamente distribuídos sem a necessidade de implantações de custosos hardwares reais.

Os cenários providos pelo CORE são emulados dinamicamente na medida em que a execução se desenvolve, possibilitando conexões de rede entre ambientes reais e emulados, permitindo a criação de diversos hosts virtuais baseados em Linux. Tanto o Servidor de Contexto como os Servidores de Borda foram emulados no CORE.

O hardware utilizado nos testes com o CORE é dotado de um processador core i5 5200U de 2.2 GHz, 8 GB de RAM DDR3L e HD 1TB 5400 RPM, com distribuição Linux (Ubuntu 16.04), por sua vez os códigos pertinentes aos Servidores de Borda, Gateways e Sensores são executados em ambiente de emulação.

5.2. Resultados e Discussões

A avaliação da proposta de *Fog Computing* para o *middleware* EXEHDA, considerou as duas situações a seguir:

- **Operação sem FOG:** a publicação dos dados contextuais dos sensores é transmitida ao Servidor de Contexto no momento da coleta, recebendo toda comunicação sobre estado crítico de uma zona de manejo e disparando atuação remota. Nesse caso não há armazenamento de eventos e dados históricos sobre a irrigação, pois nesse sentido não há suporte a registro de eventos originados pelo Servidor de Contexto.
- **Operação com FOG:** realiza uma operação de filtragem e agregação de dados contextuais, publicando a média dos valores coletados, em períodos de 1 hora. Além de qualificar o controle do ambiente, as operações de publicação dos eventos e média da tensão do solo no Servidor de Contexto possibilitam preservar o histórico e o registro de que um evento ocorreu e que foi necessária uma atuação sob o ambiente.

[2]https://www.raspberrypi.org

[3]https://www.arduino.cc/en/Reference/

[4]http://www.boeing.com.au/products-services/research-technology.page

Foram realizadas duas avaliações: (i) aquisições regulares; e (ii) leituras em intervalos reduzidos cuja descrição está a seguir.

Na primeira avaliação realizada, **aquisições regulares**, considerou-se a aquisição de dados contextuais da tensão do solo em intervalos de 5 minutos. A Tabela 4 apresenta os resultados obtidos com e sem a utilização de Fog em relação ao período de acúmulo de dados no Servidor de Contexto. Os resultados são categorizados em quatro grupos: hora; dia; semana; mês.

Tabela 4. Quantidade de dados acumulados ao longo do período

Período de envio	**Sem Fog**	**Com Fog**
1 Hora	0,044 MBytes	0,001 MBytes
1 Dia	1,066 MBytes	0,02 MBytes
7 Dias	7,462 MBytes	0,14 MBytes
30 Dias	31,968 MBytes	0,566 MBytes

Conforme pode ser visualizado na Tabela 4, com o emprego de *Fog Computing* foi obtido uma economia de aproximadamente 98% no fluxo de dados transmitidos. Esta economia é alcançada pela característica de o Servidor de Borda não enviar os dados coletados a todo momento ao Servidor de Contexto. Assim, o Servidor de Borda se torna responsável por controlar a atuação no ambiente em questão.

Na segunda avaliação realizada, **leituras em intervalos reduzidos**, foi realizada uma variação da frequência de aquisição de dados. Esta avaliação está relacionada a cenários onde é necessário preservar um rigoroso controle sobre o ambiente. Na Tabela 5 é apresentada uma comparação ao utilizar uma estratégia de *Fog Computing* aplicada à necessidade de verificação frequente às variáveis contextuais. Esta comparação é realizada com base na quantidade de dados enviada para o Servidor de Contexto, o cenário utiliza dados de coleta de um intervalo de 30 dias.

Tabela 5. Quantidade de dados contextuais transmitidos em diferentes Intervalos

Intervalo de coleta	**Sem Fog**	**Com Fog**
1 Minuto	159 MBytes	0,556 MBytes
5 Minutos	31,968 MBytes	0,556 MBytes
10 Minutos	15,984 MBytes	0,556 MBytes

Com o uso do *Fog Computing*, o Servidor de Borda calcula o valor médio dos dados coletados e publica esse valor no servidor de contexto a cada 1 hora, enviando uma quantidade menor de dados contextuais para o Servidor de Contexto, resultando em menor utilização da rede. Enquanto no ambiente sem *Fog Computing*, as publicações de dados contextuais são realizadas no momento da coleta e todos os dados são enviados.

6. Considerações Finais

A medida que a *Internet of Things* está crescendo, dispositivos embarcados, com recursos de sensoriamento, processamento, análise e comunicação, estão permitindo que objetos sejam monitorados e controlados remotamente através da infraestrutura de rede existente.

Considerando que a área de *Fog Computing* é uma área emergente no cenário internacional, se fez necessária uma busca criteriosa de trabalhos relacionados, tendo sido selecionados os mais relevantes. Empregando como referência as características dos trabalhos relacionados, foram concebidos os mecanismos empregados na arquitetura proposta, a qual atribui os conceitos de *Fog Computing* na coleta e processamento de dados contextuais. Sua operação acontece de maneira distribuída, empregando as bordas computacionais, bem como considera o emprego de regras no momento da especificação para atender as demandas de processamento contextual das diferentes aplicações.

As avaliações realizadas com o estudo de caso exploraram as funcionalidades em relação a gerência dinâmica para aquisição de dados contextuais e seu processamento distribuído. Os resultados obtidos se mostraram oportunos na redução de volume de dados transmitidos ao Servidor de Contexto, promovendo a Ciência de Contexto e, empregando as bordas computacionais de maneira distribuída.

Dentre os aspectos levantados para a continuidade do trabalho destaca-se a implementação da proposta *Fog Computing* para o EXEHDA na EMBRAPA Clima Temperado.

Agradecimentos

O presente trabalho foi realizado com apoio da CAPES (Programa Nacional de Cooperação Acadêmica - Procad) e da FAPERGS (Programa Pesquisador Gaúcho - PqG).

Referências

Aazam, M. and Huh, E.-N. (2014). Fog computing and smart gateway based communication for cloud of things. In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pages 464–470. IEEE.

Ahrenholz, J., Danilov, C., Henderson, T. R., and Kim, J. H. (2008). Core: A real-time network emulator. In *MILCOM 2008-2008 IEEE Military Communications Conference*.

APScheduler (2018). Disponível em: https://apscheduler.readthedocs.io/en/latest. Acesso em fevereiro de 2018.

Bellavista, P., Corradi, A., Fanelli, M., and Foschini, L. (2012). A survey of context data distribution for mobile ubiquitous systems. *ACM Computing Surveys (CSUR)*, 44(4):24.

Bonomi, F. (2011). Connected vehicles, the internet of things, and fog computing. In *The Eighth ACM International Workshop on Vehicular Inter-Networking (VANET), Las Vegas, USA*, pages 13–15.

Bonomi, F., Milito, R., Natarajan, P., and Zhu, J. (2014). Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pages 169–186. Springer.

Botta, A., de Donato, W., Persico, V., and Pescapé, A. (2016). Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems*, 56:684–700.

Braga, R. and Pinto, P. (2009). Alterações climáticas e agricultura. *Inovação e Tecnologia na Formação Agrícola*, 12(2):34–56.

Carriots (2018). Carriots iot application platform. Disponível em: https://www.carriots.com/. Acesso em julho de 2018.

Davet, P., Kaiser Filho, H., João, L., Xavier, L., and Carvalho, T. (2015). Consciencia de contexto na iot: uma arquitetura distribuída e escalável. *Brazilian Symposium on Computing Systems Engineering (SBESC)*.

Fremantle, P. (2015). A reference architecture for the internet of things. *WSO2 White Paper*.

Khattak, A. M., Akbar, N., Aazam, M., Ali, T., Khan, A. M., Jeon, S., Hwang, M., and Lee, S. (2014). Context representation and fusion: Advancements and opportunities. *Sensors*, 14(6):9628–9668.

Kurniawan, A. (2016). *Sparkfun ESP8266 Thing Development Workshop*. PE PRESS, Canada. An optional note.

LinkSmart (2018). Middleware for networked embedded systems. Disponível em: https://docs.linksmart.eu/. Acesso em julho de 2018.

Lopes, J. L., de Souza, R. S., Geyer, C., da Costa, C., Barbosa, J., Pernas, A. M., and Yamin, A. (2014). A middleware architecture for dynamic adaptation in ubiquitous computing. *J. UCS*, 20(9):1327–1351.

Peralta, G., Iglesias-Urkia, M., Barcelo, M., Gomez, R., Moran, A., and Bilbao, J. (2017). Fog computing based efficient iot scheme for the industry 4.0. In *Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), 2017 IEEE International Workshop of*, pages 1–6. IEEE.

Privat, G., Lemke, L., Borscia, P., and Capdevielle, M. (2016). Edge-of-cloud fast-data consolidation for the internet of things. *Proceedings of the 9th ACM Conference on Recommender Systems*.

Souza, R., Lopes, J., Geyer, C., Garcia, C., Davet, P., and Yamin, A. (2015). Context awareness in ubicomp: An iot oriented distributed architecture. In *Electronics, Circuits, and Systems (ICECS), 2015 IEEE International Conference on*, pages 535–538. IEEE.

Souza, R. S. (2017). *Um middleware para Internet das Coisas com suporte ao processamento distribuído do contexto*. Tese de doutorado em ciência da computação, Universidade Federal do Rio Grande do Sul - UFRGS, Brasil, Porto Alegre.

Stojmenovic, I. and Wen, S. (2014). The fog computing paradigm: Scenarios and security issues. In *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, pages 1–8. IEEE.

Vermesan, O. and Friess, P. (2014). *Internet of Things-From research and innovation to Market Deployment*. River Publishers.

Xively (2018). Xively. Disponível em: http://www.xively.com. Acessado em janeiro de 2018.

Zaslavsky, A., Perera, C., and Georgakopoulos, D. (2013). Sensing as a service and big data. *arXiv preprint arXiv:1301.0159*.