Applying the Internet of Things in Precision Viticulture: An Approach Exploring the EXEHDA Middleware

Leonardo João[1]
Roger Machado[1]
[1]Federal University of Pelotas
Email: {ldrsjoao, rdsmachado}
@inf.ufpel.edu.br

Verônica Tabim[2]
and Anderson Cardoso[2]
[2]Catholic University of Pelotas
Email: {veronica.tabim, anderson}
@ucpel.edu.br

João Ladislau Lopes[3]
Ana Pernas[1]
and Adenauer[12] Yamin
[3]Federal Institute Sul-rio-grandense
Email: joaolopes@cavg.ifsul.edu.br
{marilza, adenauer}@inf.ufpel.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. Regarding this scenario, the objective of this work is to present the approach designed to provide Fog Computing to the EXEHDA middleware, providing Context Awareness for the computational edge devices present in the monitored environments. A study case was carried out in the area of Precision Viticulture (PV) evaluating the proposed approach.**

***Index Terms*—Internet of Things, Fog Computing, EXEHDA middleware**

I. Introdução

A popularização crescente da Internet e a qualificação dos sistemas distribuídos, associado ao fato das tecnologias modernas potencializarem o emprego de soluções baseadas em hardware móvel, criou um cenário oportuno para a Computação Ubíqua (UbiComp). Este conceito, introduzido por Mark Weiser [1], tem como objetivo transpor as funcionalidades dos computadores convencionais (centralizados) apontando para um cenário constituído por diversos objetos inteligentes, cada um com suas funcionalidades, dotados de dispositivos computacionais distribuídos e interligados por diferentes canais de comunicação [2].

Neste cenário de milhares de dispositivos embarcados produzindo dados contextuais, a *Cloud Computing* surge como uma viabilizadora da IoT (*Internet of Things*), no sentido de prover poder de processamento e armazenamento sob demanda [3]. Porém, alguns desafios de infraestrutura são originados desta comunicação de dados entre a *Cloud* e as bordas, onde os dados são coletados. Torna-se essencial a utilização de técnicas de filtragem e fusão de dados antes de enviá-los para a *Cloud*, visando otimizar o emprego dos recursos de rede, recursos de armazenamento, e em casos de conexões com canais de baixa velocidade minimizar sua chance de sobrecarga.

Em alguns casos, também é necessária uma política de atuação sobre o ambiente Ubíquo, baseada nos dados coletados, com atuação próxima ao tempo real. Para esta proposta, em que o esforço para ciência de contexto também acontece nas bordas computacionais dá-se o nome de *Edge-of-Cloud* ou, mais usualmente, *Fog Computing* [4]. Como premissa, o *Fog Computing* prevê uma gerência autônoma dos recursos envolvidos, com regras disparadas a partir dos eventos que acontecem nos equipamentos de borda envolvidos.

Os dados coletados na infraestrutura computacional nem sempre precisam ser transmitidos para a *Cloud*. 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 [5].

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 do *Fog Computing* [6].

Diferentes desafios estão presentes no provimento de suporte para as aplicações direcionadas a IoT, considerando o paradigma de *Fog Computing*. Dentre estes, destaca-se o gerenciamento das informações coletadas através de dispositivos heterogêneos, e a interpretação destas informações considerando o contexto de interesse das aplicações [7]. 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 prover suporte de *Fog Computing* para o *middleware* EXEHDA, provendo Ciência de Contexto para os dispositivos de borda computacional presentes nos ambientes monitorados.

A principal contribuição desse trabalho é a concepção de uma arquitetura para o Subsistema de Reconhecimento de Contexto e Adaptação do EXEHDA, capacitando o mesmo a empregar os conceitos de *Fog Computing*, explorando a interoperabilidade entre as bordas computacionais. Desta forma, é diminuída a quantidade de dados transmitidos dos Servidores

de Borda para o Servidor de Contexto, ocasionando uma economia de banda e mantendo o mesmo gerenciamento do ambiente.

O texto do artigo está estruturado da seguinte forma: a seção II discute a fundamentação teórica desse trabalho, sendo apresentados os temas relacionados a Ciência de Contexto, *Fog Computing* e ao *middleware* EXEHDA. Na sequência, a seção III descreve os trabalhos relacionados. A seção IV, apresenta a abordagem desenvolvida. A seção V discute a avaliação da abordagem proposta, sendo apresentados os hardwares e softwares utilizados, e ainda, os resultados e discussões a respeito do estudo de caso. Finalmente, na seção VI são discutidas algumas contribuições alcançadas com o desenvolvimento desse trabalho e os possíveis trabalhos futuros.

II. Fundamentação Teórica

Neste capítulo são apresentados os conceitos identificados como importantes, durante a revisão de literatura em relação ao tema proposto, e que constituem a fundamentação teórica do trabalho desenvolvido.

A. Ciência de Contexto

A Ciência de Contexto refere-se a um modelo de computação no qual o sistema computacional é capaz de verificar as características do meio no qual tem interesse e, quando necessário, reagir as suas alterações [8]. No cenário da IoT, a Ciência de Contexto possui elevado significado, uma vez que existe a real necessidade do sistema tomar decisões autônomas envolvendo os inúmeros dispositivos embarcados que estão interoperando, considerando as especificações do usuário, ou aquelas inferidas pelo próprio sistema.

De acordo com [9], para a construção de sistemas cientes de contexto em ambientes altamente distribuídos, como o da IoT, alguns desafios devem ser tratados: (i) aquisição do contexto a partir de fontes heterogêneas e distribuídas; (ii) processamento dos dados contextuais adquiridos e a respectiva atuação sobre o meio físico; e (iii) disponibilização distribuída dos dados contextuais processados.

B. Fog Computing

A *Cloud Computing* é uma das tecnologias mais promissoras para a materialização da IoT, por viabilizar sistemas escaláveis ao fornecer uma elevada capacidade de armazenamento de dados e processamento [10]. Apesar desses recursos, as soluções baseadas em *cloud* necessitam de 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 [11]. Deste cenário, decorre a motivação para o uso do *Fog Computing* no *middleware* EXEHDA.

O *Fog Computing* surgiu com a intenção de abordar desafios relacionados a baixa latência, ciência de localização e confiabilidade [6]. O *Fog Computing* é implementado nas bordas computacionais, normalmente com dispositivos embarcados. Essa infraestrutura, feita com os dispositivos mais próximos das necessidades do usuário, é considerada como uma *Cloud* mais próxima do usuário, proporcionando assim baixa latência, ciência de localização e possibilidade de aprimoramento da *Quality of Service* (QoS). Deste modo, o *Fog Computing* fornece armazenamento temporário, capacidade de raciocínio, auxiliando na tomada de decisão, considerando os dados de contexto coletados.

C. Middleware EXEHDA

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 [12].

O EXEHDA é uma arquitetura baseada em eventos, gerenciada por regras, fornecendo ambiente de processamento distribuído, podendo atuar proativamente na coleta de informações contextuais do ambiente físico, bem como realizar operações remotamente sobre ele.

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.

III. Revisão de Literatura em *Fog Computing*: Principais Trabalhos

Nesta seção são 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 [13] 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

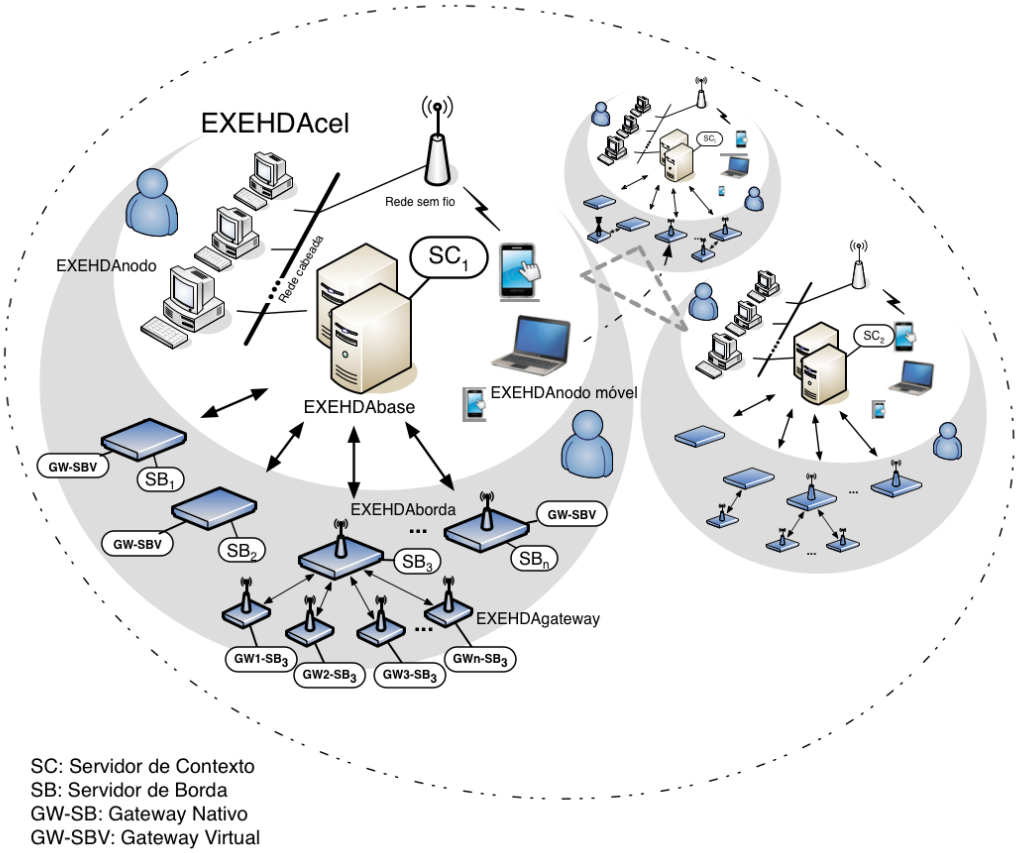

Figura 1. Ambiente Ubíquo Gerenciado pelo EXEHDA

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 [14] disponibiliza um *middleware* comercial e de código fechado, utilizando uma abordagem baseada em *Cloud* 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. A plataforma suporta diversos tipos de formatos de dados, porém não são homogeneizados automaticamente, devendo assim ser realizado um pré-processamento a fim de combiná-los [15].

LinkSmart [16] é uma plataforma de *middleware* baseada em *Service-Oriented Architecture* (SOA), desenvolvida na linguagem Java. Este *middleware* oferece suporte ao desenvolvimento de aplicações com base em informações fornecidas por dispositivos físicos heterogêneos, disponibilizando interfaces de serviços Web para controle destes dispositivos. Sua arquitetura possui três camadas principais: (i) camada de rede, responsável pela comunicação com os dispositivos; (ii) camada de serviço, responsável pelo gerenciamento de eventos, dispositivos, escalonamento de recursos, dentre outros; e (iii) camada semântica.

Carriots [17] propõe uma plataforma para aplicações em IoT que utiliza serviços de *Cloud* 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 *Cloud Computing*. A partir de sua API *RESTful*, Carriots tem o objetivo coletar e armazenar qualquer dado originado por dispositivos, utilizá-lo em seu motor de aplicações e disponibilizá-lo a seus usuários não importando o volume de dispositivos conectados.

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.

IV. Abordagem Desenvolvida

A abordagem de *Fog Computing* para o *middleware* EXEHDA herda as características de ciência de contexto, empregando uma estratégia colaborativa entre a aplicação

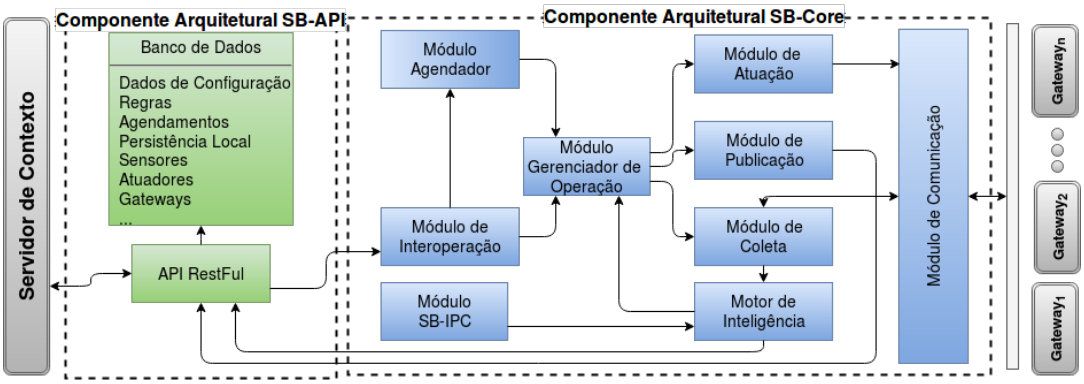

Figura 2. Módulos do Servidor de Borda.

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 da abordagem desenvolvida para o Servidor de Borda do *middleware* EXEHDA, sendo mostrados os módulos que a compõem.

A. Componente Arquitetural SB-API

O Componente Arquitetural SB-API, implementado 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 pelo Componente Arquitetural SB-API, e transmitidos 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 I são apresentados os 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*.

Tabela I
DESCRIÇÃO DAS URI DO SBWEBSERVICE

Método	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	Remove o recurso

B. Componente Arquitetural SB-Core

O Componente Arquitetural SB-Core é responsável pela atuação e processamento dos dados cadastrados no Componente Arquitetural SB-API, bem como pelo acionamento de regras sobre os eventos a serem tratados.

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

As mensagens transferidas entre cada um dos módulos utilizam-se do padrão JSON, ocorrendo um sincronismo de argumentos necessários no processamento das informações em cada etapa do Servidor de Borda. Quando o Componente Arquitetural SB-API recebe dados do Servidor de Contexto, o mesmo salva as informações no Banco de Dados, comunicando o Componente Arquitetural SB-Core sobre uma possível inserção de um novo sensor ou de um agendamento. Os agendamentos são registrados para sensores previamente cadastrados. Na Tabela II podem ser visualizadas as URI's utilizadas.

Tabela II
DESCRIÇÃO DAS URIS DO COMPONENTE ARQUITETURAL SB-CORE

Método	URI	Notifica o Core sobre:
POST	IP/sigSensor_add	novo sensor
POST	IP/sigSensor_delete	remoção de sensor
POST	IP/sigScheduler_add	novo agendamento
POST	IP/sigScheduler_delete	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) [18].

O *APScheduler* permite o cadastro de uma ação, considerando que esta 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 Coleta

Tem função de direcionar os pedidos de coleta ao Módulo de Comunicação que envia a requisição aos respectivos Gateways, sob demanda tanto do Motor de Inteligência, como do Servidor de Contexto ou das aplicações a qualquer momento. O módulo realiza a recepção assíncrona a partir das solicitações, repassando os dados no formato JSON com os parâmetros necessários para o processamento ao Motor de Inteligência.

Módulo de Atuação

Possui um funcionamento análogo ao Módulo de Coleta. O mesmo recebe informações com o ID do atuador e os correspondentes padrões de operação (tempo de duração, potência de ativação, etc.), e os envia ao Módulo de Comunicação a fim de serem interpretados e direcionados ao gateway responsável pelo efetivo tratamento.

Módulo de Publicação

Dispara as requisições de envio de informações contextuais para as demais camadas do middleware, trocando dados com o Servidor de Contexto. Considerando possíveis falhas de comunicação entre o Servidor de Borda e o Servidor de Contexto bem como eventuais atrasos da rede, é empregado uma função de Persistência Local cuja a função é realizar o armazenamento temporário das informações contextuais até que as mesmas sejam publicadas.

Motor de Inteligência

Este módulo é responsável pelo processamento das regras, as quais são armazenadas localmente e elaboradas de acordo com as necessidades da aplicação do usuário final, prioritariamente a atender os eventos críticos. Isso se deve ao fato do Servidor de Borda é geralmente alocado fisicamente próximo ao ambiente monitorado permitindo uma atuação (alertas, ativação/desativação de equipamentos eletromecânicos, etc.), independentemente de uma eventual perda de comunicação com o Servidor de Contexto. O Motor de Inteligência[2] é implementado em Python, de código aberto e sendo possível alterá-lo para uso em fins específicos.

Módulo de Comunicação

Tem a função de prover aos demais módulos da arquitetura um acesso padronizado aos recursos disponibilizados pelos Gateways no que diz respeito ao tratamento de sensores e atuadores. Gerencia as comunicações com os Gateways em uso através da API REST, por meio de protocolos que se baseiam em princípios REST como o UPnP e CoAP.

[2]https://github.com/venmo/business-rules

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 de execução. Essas URI's de acesso, podem ser visualizadas na Tabela II.

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 Inteligência.

Tabela III
DESCRIÇÃO DAS URI DO GATEWAY

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

V. AVALIAÇÃO DA ABORDAGEM PROPOSTA

A abordagem *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 *Cloud Computing* mais próxima da ocorrência dos eventos de interesse. Uma discussão detalhada da arquitetura do Servidor de Borda do EXEHDA pode ser encontrada em [19].

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 [20].

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 por meio 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.

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

A. Hardware e Software Utilizados

O Servidor de Borda foi concebido para ser disponibilizado nas instalações físicas por meio de um modelo equivalente ou superior a Raspberry PI [3] B+, usando o sistema operacional Raspbian. Os Gateways são dispositivos de baixo poder computacional, utilizados na comunicação entre os Sensores e Servidores de Borda. Considerando o baixo consumo energético e o emprego de uma plataforma de software disseminada, para prototipação dos Gateways utilizou-se o processador ESP8266-12 [21], um dispositivo com suporte a comunicação WI-FI e programável através do framework de desenvolvimento da plataforma Arduino[4].

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) [22]. Este framework *open-source* desenvolvido pela *Boeing Research and Development Division* (BR&T)[5] 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. O CORE pode ser usado via GUI e/ou Python Scripting, geralmente a GUI é utilizada para desenhar os nós interconetando dispositivos pela interface gráfica; e módulos em python são utilizados para configurar e instanciar nós e redes. 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 de quatro núcleos, 8 GB de memória RAM DDR3L e HD 1TB 5400 RPM, com Sistema Operacional Linux (Ubuntu 16.04), por sua vez os códigos pertinentes aos Servidores de Borda, Gateways e Sensores são executados em ambiente de emulação.

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

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

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

B. Resultados e Discussões

Para realização do estudo de caso foi considerada uma distribuição de Servidores de Borda, Gateways e sensores ao longo da área típica de um vinhedo. Considerando 6 Zonas de Manejo, onde cada uma é composta por:

- **Servidor de Borda:** cada Zona de Manejo é gerenciada por um Servidor de Borda sendo responsável pela centralização dos dados contextuais coletados e pela gerência autônoma de irrigações a serem realizadas;
- **Gateway:** cada Servidor de Borda gerencia 6 Gateways dentro da sua Zona. A comunicação com o Servidor de Borda ocorre via WIFI e com os sensores fisicamente, utilizando fios na troca de dados. Por isso, necessitam de uma distribuição maior ao longo da Zona de Manejo;
- **Sensoriamento:** cada Gateway coordena a operação 5 sensores de tensão de solo (tensiômetros).

Com a organização acima tem-se no estudo de caso 180 sensores, 36 Gateways, 6 Servidores de Borda e 1 Servidor de Contexto sendo emulados através do CORE. A distribuição dos Gateways e Servidores de Borda ao longo do Vinhedo é feita conforme apresentado na Figura 3. Essa figura resume o esforço de concepção no ambiente provido pelo CORE para acomodar de forma distribuída os Servidores de Borda e os Gateways conectados aos mesmos. Através da interface gráfica é possível ver as interconexões da infraestrutura de FOG, bem como ter acesso aos seus aspectos operacionais clicando sobre os diferentes ícones.

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.

Foram realizadas três avaliações: (i) aquisições regulares; (ii) leituras em intervalos reduzidos; (iii) Escalabilidade dos Sensores Empregados 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 IV apresenta os resultados obtidos com e sem a utilização de Fog em relação

Figura 3. Zonas de Manejo no Estudo de Caso

ao período de acúmulo de dados no Servidor de Contexto. Os resultados obtidos foram agrupados em quatro intervalos de tempo: hora; dia; semana; e mês.

Tabela IV
QUANTIDADE DE DADOS ACUMULADOS AO LONGO DO PERÍODO

Período de envio	**Sem Fog**	**Com Fog**
1 Hora	0,7 MBytes	0,004 MBytes
1 Dia	19 MBytes	0,17 MBytes
7 Dias	133 MBytes	1,19 MBytes
30 Dias	572 MBytes	4,39 MBytes

A redução do volume de dados observada na Tabela IV possui ampla importância no envio de dados para o Servidor de Contexto por meio de internet móvel, com franquia limitada e/ou de baixa qualidade, comum em ambientes rurais como no cenário analisado. Com o emprego de *Fog Computing* foi obtido uma economia de aproximadamente 99% no fluxo de dados transmitidos, no cenário analisado. Esta economia é alcançada pela característica de o Servidor de Borda não transmitir 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. Essa característica potencializa a escalabilidade a nível de borda, possibilitando a conexão de mais sensores interconectados aos Servidores de Borda.

Na segunda avaliação realizada, Leituras em Intervalos Reduzidos, aumentou-se a frequência de aquisição de dados. Esta avaliação está relacionada a cenários onde é necessário preservar um rigoroso controle sobre o ambiente. Porém intervalos pequenos na aquisição de informações contextuais podem constituir um problema operacional para arquiteturas que utilizam conceito de inteligência centralizada, uma vez que todos dados contextuais devem ser transmitidos para estes locais e, principalmente, os que necessitam de uma transmissão via internet. Desta forma, podendo sobrecarregar a rede existente ou limitar a banda disponível para outras operações de interesse do usuário, causando assim um dilúvio de dados ou congestionamento do canal de transmissão.

Na Tabela V 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 V
QUANTIDADE DE DADOS CONTEXTUAIS TRANSMITIDOS EM DIFERENTES INTERVALOS

Intervalo de coleta	**Sem Fog**	**Com Fog**
1 Minuto	2877 MBytes	4,39 MBytes
5 Minutos	572 MBytes	4,39 MBytes
10 Minutos	298 MBytes	4,39 MBytes

Com o emprego de *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 transmitidos ao Servidor de Contexto.

Na terceira avaliação realizada, Escalabilidade dos Sensores Empregado, é considerado a avaliação de escalabilidade nas bordas computacionais, realizando testes na distribuição de sensores para os Gateways e, por associação por Servidor de Borda. O resultado do volume de dados transmitido alterando-se a quantidade de sensores pode ser observado na Tabela VI.

Tabela VI
QUANTIDADE DE DADOS CONTEXTUAIS TRANSMITIDOS EM DIFERENTES INTERVALOS

Intervalo de coleta	**Sem Fog**	**Com Fog**
1 Sensor	115 MBytes	4,39 MBytes
3 Sensores	350 MBytes	4,39 MBytes
5 Sensores	572 MBytes	4,39 MBytes

De acordo com a tabela VI, foi observado que com o uso da *Fog Computing* independentemente do número de sensores presentes no ambiente, a quantidade de dados enviados ao Servidor de Contexto permanece a mesma. Isto deve-se ao fato do Servidor de Borda processar os dados a nível de borda e, utilizar de agregação de eventos de leitura, onde a média os dados é realizada, e sendo propagada somente esta média para o Servidor de Contexto, sendo possível assim a redução dos dados transmitidos.

Além disso, a estratégia de distribuição adotada permite expandir o número de dispositivos conforme o aumento do volume de informações geradas, evitando assim possíveis sobrecargas nos dispositivos envolvidos. Isso significa que se lavoura fosse expandida e aumentasse a quantidade de Zonas de Manejo, mais Servidores de Bordas e Gateways seriam utilizados, de modo que a quantidade de informações a ser tratada por cada dispositivo possa ser mantida sobre controle.

VI. CONCLUSÕES

A revisão de literatura na área apontou que a estratégia de *Fog Computing*, baseada em distribuição de eventos, mostra-se promissora para promover a descentralização dos procedimentos de processamento contextual, expandindo as atuais soluções baseadas na *Cloud Computing*.

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. É possível notar que o volume de dados enviado ao Servidor de Contexto é reduzido significativamente, sem modificar a operação do ambiente, ou seja, as atuações e processos são os mesmos no ambiente com uso da *Fog Computing* ou sem. 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 Empresa Brasileira de Pesquisa Agropecuária, EMBRAPA Clima Temperado.

REFERÊNCIAS

[1] M. Weiser, "Some computer science issues in ubiquitous computing," *Communications of the ACM*, vol. 36, no. 7, pp. 75–84, 1993.

[2] X. Wang, S. Ong, and A. Nee, "A comprehensive survey of ubiquitous manufacturing research," *International Journal of Production Research*, pp. 1–25, 2017.

[3] M. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things: A survey," *Internet of Things Journal, IEEE*, vol. 3, no. 1, pp. 70–95, 2016.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513

[5] J. Ni, K. Zhang, X. Lin, and X. Shen, "Securing fog computing for internet of things applications: Challenges and solutions," *IEEE Communications Surveys & Tutorials*, 2017.

[6] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 Workshop on Mobile Big Data*. ACM, 2015, pp. 37–42.

[7] V. Gazis, A. Leonardi, K. Mathioudakis, K. Sasloglou, P. Kikiras, and R. Sudhaakar, "Components of fog computing in an industrial internet of things context," in *Sensing, Communication, and Networking-Workshops (SECON Workshops), 2015 12th Annual IEEE International Conference on*. IEEE, 2015, pp. 1–6.

[8] J. Augusto, A. Aztiria, D. Kramer, and U. Alegre, "A survey on the evolution of the notion of context-awareness," *Applied Artificial Intelligence*, vol. 31, no. 7-8, pp. 613–642, 2017.

[9] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini, "A survey of context data distribution for mobile ubiquitous systems," *ACM Computing Surveys (CSUR)*, vol. 44, no. 4, p. 24, 2012.

[10] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, and M. Nemirovsky, "Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing," in *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2014 IEEE 19th International Workshop on*. IEEE, 2014, pp. 325–329.

[11] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards proactive mobility-aware fog computing," p. 72, 05 2017.

[12] L. Xavier, H. Kaiser, P. Davet, E. Abreu, W. Parreira, and A. Yamin, "Instrumentação na iot: uma abordagem autônoma e distribuída," *Simpósio Brasileiro De Telecomunicações - SBrT*, 2016.

[13] P. Fremantle, "A reference architecture for the internet of things," *WSO2 White Paper*, 2015.

[14] N. Sinha, K. E. Pujitha, and J. S. R. Alex, "Xively based sensing and monitoring system for iot," in *2015 International Conference on Computer Communication and Informatics (ICCCI)*, Jan 2015, pp. 1–6.

[15] A. Zaslavsky, C. Perera, and D. Georgakopoulos, "Sensing as a service and big data," *arXiv preprint arXiv:1301.0159*, 2013.

[16] LinkSmart, "Linksmart device integration and abstraction services," 2018, disponível em: https://docs.linksmart.eu/pages/viewpage.action?pageId=7667778. Acesso em abril de 2018.

[17] C. by ALTAIR, "Carriots - internet of things platform," 2018, disponível em: https://www.carriots.com/what-is-carriots. Acesso em abril de 2018.

[18] APScheduler, 2018, disponível em: https://apscheduler.readthedocs.io/en/latest. Acesso em fevereiro de 2018.

[19] R. S. Souza, "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, 2017.

[20] R. Braga and P. Pinto, "Alterações climáticas e agricultura," *Inovação e Tecnologia na Formação Agrícola*, vol. 12, no. 2, pp. 34–56, 2009.

[21] A. Kurniawan, *Sparkfun ESP8266 Thing Development Workshop*. Canada: PE PRESS, 2016, an optional note.

[22] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "Core: A real-time network emulator," in *MILCOM 2008-2008 IEEE Military Communications Conference*, Oct. 2008.