IMAIoT – Infrastructure Monitoring Agent for IoT: Um Agente Monitor de Infraestruturas para Ambientes de IoT

Alexandre Heideker, Dener Ottolini, Ivan Zyrianoff, João Kleinschmidt, Carlos Kamienski

Universidade Federal do ABC (UFABC)

{alexandre.heideker, dener.silva, ivan.dimitry, joao.kleinschmidt, cak}@ufabc.edu.br

Abstract. *The concept of Internet of Things (IoT) comes with a large number of devices connected to the Internet, including urban, industrial and agriculture environment. Managing and monitoring these devices, whether virtual or physical, across multiple hardware and software platforms, is a major challenge. There are market solutions but for specific domain and platforms, overall closed and not customizable. We introduce the IMAIoT, an infrastructure monitoring tool that uses the high scalable IoT's protocol and architecture to publish its metrics. The tool's versatility allows to monitor from real machines in a datacenter to small devices such as fog computing node.*

Resumo. *O conceito Internet da Coisas (IoT) é caracterizado pela presença de um número muito grande de dispositivos ligados à Internet, o que inclui ambientes urbanos, industriais e de agricultura. O gerenciamento e monitoramento destes dispositivos, virtuais ou físicos, em múltiplas plataformas de hardware e software, representa um grande desafio. Há soluções disponíveis no mercado, porém, específicas para certos domínios e plataformas e, em sua maioria, proprietárias e pouco personalizáveis. Apresentamos o IMAIoT, uma ferramenta para monitoramento de infraestruturas que utiliza protocolos e arquitetura altamente escalável de IoT para disponibilizar suas métricas. A versatilidade da ferramenta permite monitorar desde máquinas reais em um datacenter até dispositivos com como nós de computação em névoa.*

1. Introdução

O conceito de Internet das Coisas (IoT), e sua ampla adoção, representa um grande desafio ao projeto e gerenciamento de infraestruturas de telecomunicação, seja pela presença de novos protocolos de comunicação, seja pela escala desse desafio frente ao número crescente de dispositivos conectados ou pela grande heterogeneidade de arquiteturas destes dispositivos.

Envolvidos nessa onda tecnológica, estão conceitos como: Computação em Nuvem, Computação em Névoa, plataformas de comunicação sem fio de baixo consumo de energia, como LoRaWAN [Sornin *et al.*, 2016], adoção de diferentes arquiteturas de computação, como ARM, RaspberryPi[1], entre outras, além das técnicas de integração e gerenciamento de infraestruturas virtualizadas, como: Virtualização de Funções de Rede

[1] https://www.raspberrypi.org acesso em 10/03/2019.

– NVF [Cotroneo *et al*., 2014] e Redes Definidas por Software – SDN [Mckeown *et al*., 2008]. Em comum, todas estas tecnologias apresentam direta ou indiretamente a necessidade de coletar métricas de software e hardware para a tomada de decisão de gerenciamento e monitoramento. Outro aspecto importante a ser considerado é a heterogeneidade destes ambientes, inclusive relacionada ao conteúdo legado, seja de hardware ou software.

As diferentes ferramentas de gestão de infraestrutura utilizadas atualmente utilizam protocolos específicos, como o SNMP, além de aplicações fortemente ligadas a sua respectiva plataforma, tornando inviável a coexistência entre diversos fornecedores. Nota-se também a baixa capacidade de customização, considerando a forte presença de soluções de código fechado, além da dificuldade de realizar este monitoramento através de diferentes domínios de rede, como é o caso da arquitetura de Computação em Névoa – FOG [Al-Fuqaha *et al*., 2015].

O IMAIoT – *Infrastruture Monitoring Agent for IoT* foi desenvolvido justamente para realizar esta tarefa, independente da plataforma de fornecimento de infraestrutura. Utilizando a arquitetura *Publish/Subscribe*, as métricas podem ser utilizadas para diversos fins, desde a gestão e monitoramento da infraestrutura, até a divisão de carga de trabalho em processos de elasticidade de aplicações e de infraestrutura (como em plataformas de NFV).

No IMAIoT, as métricas obtidas dos dispositivos monitorados podem ser acessadas de forma indireta, com o uso de um *broker* para IoT, ou diretamente, através de um *socket* TCP. O *broker* na arquitetura de IoT funciona como um grande armazém de estados dos dispositivos a ele associados, recebendo atualizações destes e notificando os seus subscritos. A ferramenta também pode ser utilizada para realização de experimentos e avaliações de desempenho, já que conta com a opção de geração de registros históricos com as métricas monitoradas para posterior análise em uma operação totalmente autônoma. Escrito em C++, a ferramenta pode ser facilmente portada para diversas plataformas de forma rápida, com o mínimo de *overhead* no sistema hospedeiro, ocupando cerca de 30Kbytes de sua memória RAM em sua compilação para RaspberryPi por exemplo.

Finalmente, uma aplicação para visualização das métricas graficamente (*dashboard*) é apresentada como exemplo, utilizando simples programação em JavaScript, demonstrando a versatilidade no uso do IMAIoT.

A seção 2 apresenta a motivação do desenvolvimento da ferramenta e os trabalhos correlatos. Na seção 3 a arquitetura da ferramenta é apresentada. A seção 4 apresenta um exemplo de aplicação e a estrutura da demonstração e, finalmente na seção 5 as conclusões são apresentadas.

2. Motivação e Trabalhos Correlatos

As tecnologias de virtualização e o movimento de migração de recursos computacionais com a adoção da computação em nuvem, a computação como serviço e sua onipresença, geram uma pressão cada vez maior por ferramentas e técnicas de gerenciamento e orquestração.

Esta demanda pode ser observada na academia com um grande número de trabalhos que avaliam desempenho utilizando métricas de hardware e software, e neste

ambiente, precisão e homogeneidade de técnicas para a coleta de métricas é essencial. Em [Coutinho *et al*., 2015] os autores apresentam uma revisão de técnicas de elasticidade em Computação em Nuvem onde métricas como CPU, uso médio de rede, entre outras são elementos chave para promover o *auto scaling*. Avaliações de desempenho como vistas em [Heideker; Zyrianoff; Kamienski, 2018; Zyrianoff *et al*., 2017, 2018] mostram que monitorar os recursos de forma precisa é essencial para certificar as conclusões destes estudos.

No caso da tecnologia NFV, a detecção de gargalos em Funções de Rede Virtualizadas é essencial para garantia a qualidade de serviço exigida nesse ambiente, e a importância do uso de métricas confiáveis é explorado em [Khalid *et al*., 2016]. Em propostas como a de [Miotto *et al*., 2019], a figura do *Metric Collector* traduz-se em um agente que fornece as informações sobre o estado da máquina virtual.

Em geral as plataformas de Computação em Nuvem como a Amazon AWS, Google Cloud, Microsoft Azure, OpenStack, entre outras, possuem APIs que fornecem estas informações, contudo, estas informações limitam-se à sua respectiva plataforma e domínio de rede. Outras ferramentas independentes da plataforma de nuvem estão disponíveis, tais como: SolarWinds[2], PRTG[3], Nagios[4], OpsView[5], Zenoss[6], entre outros. Em comum, todas estas ferramentas são pagas. Apesar de o Nagios ser *opensource*, seus recursos avançados e ferramentas de administração são pagas. Ferramentas como o Prometheus[7] e o Graphite[8], apesar de serem *opensource*, são desenvolvidas em linguagens de programação de alto nível e possuem uma arquitetura baseadas em registro local, o que torna sua utilização em ambientes com poucos recursos computacionais inviável.

O IMAIoT diferencia-se destas ferramentas por tornar o agente independente da interface de administração, ou seja, o agente é autônomo, em um modo de operação onde as métricas são armazenadas em arquivos locais para a posterior análise. A configuração do IMAIoT no modo *context broker*, onde suas métricas são reportadas no FIWARE Orion, também representa um diferencial por unificar o tratamento da informação em sistemas de IoT em um mesmo protocolo e arquitetura, ou seja, a informação de um sensor ou atuador é tratada da mesma forma e pelo mesmo canal que a informação de monitoramento da infraestrutura.

O projeto FIWARE[9] tem por objetivo a construção de um ecossistema de código aberto para criação de soluções inteligentes através de uma API padronizada. Apesar de não se limitar ao conceito de IoT, o FIWARE vem ganhando notoriedade em projetos voltados à Cidades Inteligentes e IoT. Neste contexto, o FIWARE Orion apresenta-se como sua mais representativa contribuição. O FIWARE Orion é um *context broker*, uma espécie de armazém de estados, onde dispositivos de IoT ou outros softwares se conectam, em uma operação de *publish* – quando estes elementos publicam dados no

---

[2] https://www.solarwinds.com/server-application-monitor/ acesso em 13/03/2019.

[3] https://www.paessler.com/prtg acesso em 18/03/2019.

[4] https://www.nagios.com acesso em 11/02/2019.

[5] https://www.opsview.com/downloads acesso em 18/03/2019.

[6] https://www.zenoss.com acesso em 18/03/2019.

[7] https://prometheus.io/ acesso em 12/04/2019.

[8] https://graphiteapp.org/ acesso em 12/04/2019.

[9] https://www.fiware.org acesso em 21/03/2019.

broker – ou *subcriber* – quando dispositivos de IoT ou outros softwares são notificados quando há modificações nas informações armazenadas, que são organizadas em tópicos e sub tópicos.

3. Arquitetura do IMAIoT

A Figura 1 mostra a arquitetura geral do IMAIoT, que utiliza basicamente quatro *threads* em sua execução. O primeira é a *thread* responsável por coletar as diversas métricas de hardware e software, preenchendo uma estrutura de dados para a posterior consulta. A segunda thread é responsável por prover um socket TCP para consulta direta ao estado do agente, fornecendo uma estrutura de dados do tipo JSON com as métricas coletadas ao aplicativo de consulta. A terceira *thread* é utilizada para gerar um arquivo de registro histórico (*log*) na máquina local no caso de experimentos de avaliações de desempenho ou simplesmente como histórico local. É possível configurar o formato deste arquivo como texto separado por ponto e virgula ou JSON. Finalmente a quarta *thread* é responsável por publicar as métricas coletadas no FIWARE Orion.

A opção pelo desenvolvimento em múltiplas *threads* se deve ao fato de o IMAIoT permitir tempos assíncronos de amostragem de métricas e publicação/log, ou seja, a amostragem das métricas poderá ser feia em um intervalo de tempo diferente do intervalo de registro das mesmas, além de mitigar o atraso provocado tanto na coleta das métricas como nas diversas formas de publicação das mesmas em favor da precisão no tempo absoluto (*timestamp*) da amostragem. Vale ressaltar que muitas das plataformas de hardware onde a ferramenta pode ser utilizada, por exemplo um RaspberryPi, possuem limitações na velocidade de gravação de mídias (cartão de memória) além de sobrecargas causadas por picos de uso das aplicações.

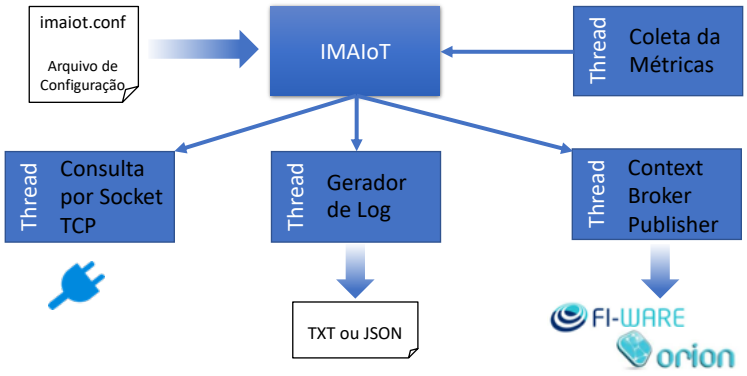

Figura 1: Arquitetura Geral do IMAIoT.

Como há um grande número de variáveis para modelar o comportamento do IMAIoT, optou-se por realizar esta configuração através de um arquivo de texto plano com o nome das variáveis seguidos o símbolo de igualdade (=) e seus respectivos valores atribuídos. A Tabela 1 descreve as principais variáveis do arquivo `imaiot.conf`.

Entre as diversas variáveis apresentadas na Tabela 1, vale destacar a possibilidade de não só coletar as métricas do sistema operacional hospedeiro, como também de processos em execução, mas em especial de containers Docker[10] - esta característica

[10] https://www.docker.com/ acesso em 25/02/2018.

permite a obtenção de métricas em máquinas que hospedam microserviços que utilizam o Docker como ambiente de virtualização, atuando como sensor em processos de elasticidade. O resultado final é não só o monitoramento da máquina real ou virtual, como também das aplicações envolvidas no *Service Function Chaining* - SFC.

Tabela 1: Principais Variáveis de Configuração do IMAIoT.

Variável	Descrição
`NodeUUID = urn:ngsi-ld:999`	[texto] UUID utilizado no Orion Context Broker para identificar unicamente o agente.
`KindOfNode = LoRaGateway`	[texto] Identifica o tipo de nó monitorado para simplificar a consulta do mesmo na plataforma de IoT
`SamplingTime = 5`	[inteiro] Intervalo de coletas das métricas (segundos)
`LogMode = 0`	[boleano] Registra em arquivo local as métricas
`LogType = txt`	[txt | json] Formato do arquivo de log, se texto separado por ponto e virgula (txt) ou JSON
`LogFileName = imaitlog.txt`	[texto] Nome do arquivo de log.
`LogIntervall = 5`	[inteiro] Intervalo de registro das métricas (segundos)
`ServerMode = 1`	[boleano] Ativa o socket TCP para consulta
`ServerPort = 5999`	[inteiro] Porta utilizada para a o socket TCP
`OrionMode = 0`	[boleano] Ativa a publicação das métricas coletadas no Orion Context Broker
`OrionHost = http://hostname`	[texto] URL para o Orion Context Broker
`OrionPort = 1026`	[inteiro] Porta para o Orion Context Broker
`OrionPublisherTime = 30`	[inteiro] Intervalo entre as publicações de métricas (segundos)
`DockerStat = 0`	[boleano] Coleta de estatísticas de containers dockers
`DockerNames = *`	[texto] Lista de containers, separados por espaço, ou asterisco (*) para todos os containers em execução
`ProcessNames = bash apache`	[texto] Lista de nomes de processos do sistema operacional separados por espaço.

Um exemplo da versatilidade da ferramenta pode ser observado na Figura 2, onde o agente IMAIoT pode ser instalado em múltiplos equipamentos, desde gateways LoRaWAN, dispositivos do tipo RaspberryPi ou servidores de aplicações em datacenters locais, até em instâncias em provedores de nuvem pública. Estas métricas podem ser inclusive publicadas em uma instância do Orion Context Broker rodando na própria infraestrutura monitorada. Neste exemplo, aplicações específicas podem ser desenvolvidas para consultar os dados do *context broker*, não só apresentando as métricas em sua forma bruta, como utilizando algoritmos diversos para obtenção de informação de alto nível do sistema como um todo.

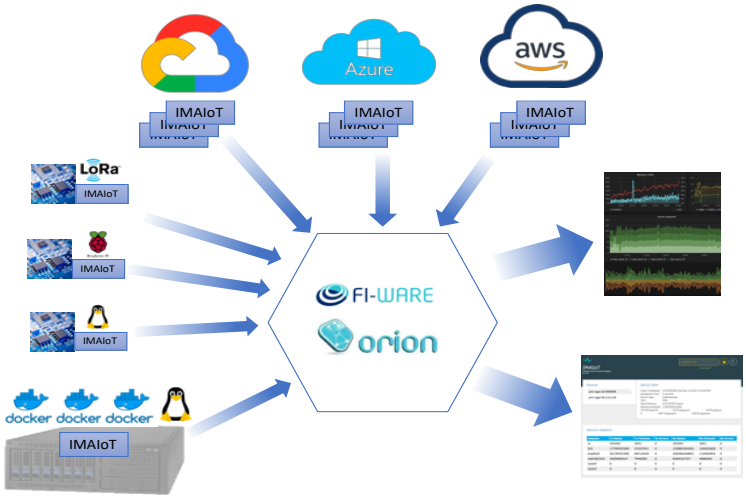

Figura 2: Exemplo de Aplicação com Múltiplos Agentes IMAIoT.

3.1 Visualização e Utilização das Métricas

A arquitetura do IMAIoT permite a obtenção das métricas coletadas através do arquivo de registro histórico, por conexão direta ao socket TCP e/ou pela consulta ou subscrição ao tópico publicado no Orion Context Broker. Assim como exemplificado na Figura 2, o repositório do IMAIoT contém uma aplicação web exemplo que efetua a conexão ao *context broker* indicado, coletando as métricas de todos os dispositivos nele publicados, finalmente exibindo-as de forma amigável ao usuário. Esta conexão ao *context broker* é realizada através da API REST do próprio Orion.

A Figura 3 mostra a interface web desenvolvida como exemplo, sendo composta de apenas um arquivo HTML contendo definições de estilo em CSS e uma aplicação em JavaScript responsável por realizar as chamadas à API REST e exibir os dados na página HTML. Esta interface pode ser adaptada e/ou expandida para atender a aplicação desejada, inclusive com atualização automática das métricas em tela. Neste exemplo, existem três agentes publicando métricas no Orion (`urn:ngsi-ld:999999`, `urn:ngsi-ld:111119` e `urn:ngsi-ld:6652`).

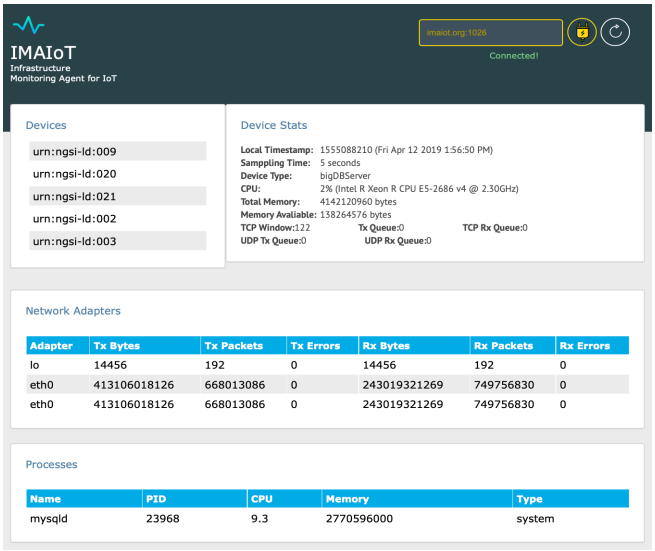

Figura 3: Interface Gráfica em um Exemplo de Aplicação Web para o IMAIoT.

Outras funcionalidades podem ser implementadas diretamente no Orion, como por exemplo o FIWARE Quantumleap, para armazenamento de séries temporais, o que permite não só a obtenção de métricas instantâneas dos agentes IMAIoT, como também o histórico de suas respectivas métricas, sem a necessidade de implementar novas aplicações ou funcionalidades no agente – basta configurar a plataforma de IoT para realizar esta tarefa de forma autônoma.

O repositório para o código fonte do IMAIoT, assim como o manual de instalação e aplicação de exemplo estão disponíveis em um repositório público do GuitHub (https://github.com/heideker/IMAIoT). Um vídeo com o procedimento de instalação e testes está disponível na plataforma YouTube (https://youtu.be/IPe_KVRKbtE).

4. Descrição da Demonstração

A demonstração será realizada utilizando um conjunto de máquinas reais e virtuais onde o agente IMAIoT estará instalado. Serão exploradas as diversas formas de consulta das métricas, observando os arquivos de log gerados, consultas em terminal via `telnet` ao *socket* TCP e consultas via `curl` ao Orion Context Broker. Paralelamente, a interface gráfica de exemplo será utilizada para ilustrar as métricas obtidas em uma aplicação usuária. O website do projeto IMAIoT (http://www.imaiot.org) contém também um exemplo de uso real da ferramenta monitorando alguns servidores físicos e virtualizados e RaspberryPi. Neste exemplo a interface da Figura 3 foi customizada para uma conexão fixa a um Orion específico além de uma figura com a topologia monitorada.

A primeira demonstração será composta pela execução do agente IMAIoT na máquina hospedeira, explorando brevemente algumas configurações, como o `DebugMode` e o `LogIntervall`. Em seguida, a demonstração será realizada efetuando o login em uma das máquinas/dispositivos monitorados e, através do uso de ferramentas do próprio sistema operacional, como o `tail`, inspecionar em tempo real o conteúdo de um arquivo de log em formato texto.

Na sequência, a mesma demonstração é realizada em uma outra máquina/dispositivo, configurada para exibir o log em formato JSON. Explorando o *socket* TCP, utilizando a ferramenta `telnet`, será observada a consulta ao agente de forma direta. Para explorar a ferramenta configurada para uso do Orion Context Broker, será utilizada a ferramenta `curl` para efetuar diversas consultas ao Orion, aplicando filtros para obter as métricas utilizando as consultas via API REST.

Finalmente a aplicação web exemplo será exibida demonstrando a versatilidade da plataforma na criação de aplicações para o usuário final ou administrador da infraestrutura. A demonstração utilizará os equipamentos para exibição disponíveis como projetores e/ou TVs além de dispositivos próprios de IoT (e.g. Raspberry Pi) e máquinas em infraestruturas remotas acessíveis por Internet.

5. Conclusão

A demanda por ferramentas de gerenciamento de infraestrutura crescerá na mesma proporção da adoção das tecnologias de IoT. Acredita-se que a ferramenta IMAIoT não só representa uma boa alternativa ao uso de ferramentas proprietárias e fortemente acopladas às plataformas específicas, como apresenta-se versátil o suficiente para a expansão de suas funcionalidades, bem como portável para outras arquiteturas.

Como trabalhos futuros, espera-se a inclusão da habilidade de interação do agente com a infraestrutura monitorada, permitindo a atuação na mesma, assim como a inclusão de protocolos de segurança para acesso ao agente, garantindo segurança e privacidade ao sistema.

6. Agradecimentos

Essa pesquisa foi parcialmente financiada pelo projeto SWAMP [Kamienksi et. al. 2019], uma colaboração entre Brasil e União Europeia.

Referências

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015) "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications". *IEEE Communications Surveys and Tutorials*, v. 17, n. 4, p. 2347–2376, 2015.

Cotroneo, D., De Simone, L., Iannillo, A. K., Lanzaro, A., Natella, R., Fan, J., & Ping, W. (2014) "Network function virtualization: Challenges and directions for reliability assurance." 2014 IEEE International Symposium on Software Reliability Engineering Workshops. IEEE.

Coutinho, E. F., de Carvalho Sousa, F. R., Rego, P. A. L., Gomes, D. G., & de Souza, J. N. (2015) "Elasticity in cloud computing: a survey." annals of telecommunications-annales des télécommunications 70.7-8: 289-309.

Heideker, Alexandre, Ivan Zyrianoff, and Carlos Kamienski. (2018) "Compreendendo o Desempenho de Serviços Encadeados Virtuais de Redes." Anais do XVI Workshop em Clouds e Aplicações (WCGA-SBRC 2018). Vol. 16. SBC.

Kamienski, C., Soininen, J.P., Taumberger, M., Dantas, R., Toscano, A., Cinotti, T.S., Maia, R.F. Torre Neto, A., "Smart Water Management Platform: IoT-Based Precision Irrigation for Agriculture", Sensors, 19, p.276, Janeiro 2019.

Khalid, J., Coatsworth, M., Gember-Jacobson, A., & Akella, A. (2016) "A standardized southbound API for VNF management." Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization. ACM.

McKeown, Nick, et al. (2008) "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM Computer Communication Review 38.2: 69-74.

Miotto, G., Luizelli, M. C., da Costa Cordeiro, W. L., & Gaspary, L. P. (2019) "Adaptive placement & chaining of virtual network functions with NFV-PEAR." Journal of Internet Services and Applications 10.1:3.

Sornin, A., Luis, M., Eirich, T., Kramp, T. e Hersent, L. (2016) "LoRaWAN ™ Specification Version 1.0.2".

Zyrianoff, I., Borelli, F., Heideker, A., Biondi, G., & Kamienski, C. (2017) "Compreendendo o Desempenho de Gerenciadores de Contexto para Internet das Coisas." 9o Simpósio Brasileiro de Computação Ubíqua e Pervasiva. SBC.

Zyrianoff, I., Borelli, F., Biondi, G., Heideker, A., & Kamienski, C. (2018) "Scalability of Real-Time IoT-based Applications for Smart Cities." 2018 IEEE Symposium on Computers and Communications (ISCC). IEEE.