SmartFogLB: Arquitetura de Balanceamento de Carga em Fog Computing *

Éder P. Pereira[1], Ivânia A. Fischer[1], Roseclea D. Medina[1], Edson L. Padoin[2]

[1]Universidade Federal de Santa Maria (UFSM) - Santa Maria - RS - Brasil

epereira@inf.ufsm.br, ivaniafischer@redes.ufsm.br, roseclea@inf.ufsm.br

[2]Universidade Reg. do Noroeste do Estado do Rio G. do Sul (UNIJUI) - Ijuí - RS - Brasil

padoin@unijui.edu.br

Resumo. *O paradigma de Computação em Névoa tem ganhado atenção nos últimos anos como um recurso para economia de largura de banda e redução de tempos de resposta nas aplicações de Internet das Coisas (IoT). Trata-se de uma camada virtualizada e intermediária entre servidores na nuvem e dispositivos IoT, normalmente localizada na borda da rede. Este trabalho propõe um modelo arquitetural de um balanceador de carga em ambientes de Computação em Névoa, cujos objetivos são: redução de nós da névoa sobrecarregados e desbalanceados, bem como, abstrair do dispositivo final IoT as falhas que ocorrem nos nós que a compõem.*

1. Introdução

A INTERNET DAS COISAS (do inglês, Internet of Things - IoT) tem ganhado bastante atenção nos últimos anos, devido ao seu rápido crescimento nas mais diversas áreas de aplicação a qual é inserida. Estima-se IoT como a grande oportunidade de mercado, e que em 2020, mais de 212 bilhões de 'coisas' estarão conectadas gerando ou consumindo informações, e que, em 2022, 45% de todo o tráfego de internet será entre máquinas autônomas [Al-Fuqaha et al. 2015].

Neste contexto, a Computação em Névoa (do inglês Fog Computing) vem para resolver 2 principais problemas gerados pela grande quantidade de dispositivos IoT. O primeiro, diz respeito ao consumo de largura de banda de internet, pois, como consequência da grande quantidade de dispositivos gerando informações e enviando-as para a nuvem, os links *wan* tendem a ficar sobrecarregados. O segundo, diz respeito ao tempo de resposta, uma vez que aplicações IoT sensíveis à latência como carros conectados, aplicações de realidade aumentada, *streaming* de vídeo, dentre outras, podem sofrer atrasos até o dado percorrer o caminho em direção à nuvem e ser tratado [Stojmenovic and Wen 2014].

Como a Computação em Névoa é formada por vários nós, os quais cooperam entre si, faz-se necessário o gerenciamento dos mesmos, bem como, a utilização correta dos recursos computacionais alocados em cada dispositivo. O balanceamento de carga entre os nós da névoa, proposto neste trabalho, é um recurso fundamental para cumprir esta tarefa, uma vez que o mesmo possui uma visão holística da infraestrutura, assim como informações detalhadas sobre os recursos computacionais que estão à sua disposição, e ainda possui o conhecimento da saúde dos nós da névoa para eventuais aplicações de políticas de balanceamento de carga. Portanto, este trabalho apresenta uma proposta de um balanceador de carga para a Computação

*Trabalho desenvolvido com recursos do edital MCTIC/CNPq - Universal 28/2018 sob número 436339/2018-8 and CAPES sob número 1765322.

em Névoa, cujo objetivo é a melhor distribuição de carga entre seus nós, consequentemente mitigar o desbalanceamento, além de ocultar falhas de seus nós dos sensores das aplicações da IoT. O restante do trabalho está organizado da seguinte forma: a Seção 2 discute os trabalhos relacionados. A Seção 3 apresenta a proposta do balanceador de carga, seguido da seção 4 com considerações finais e trabalhos futuros.

2. Trabalhos Relacionados

Diante dos atuais desafios na INTERNET DAS COISAS, diferentes propostas têm sido apresentadas, cujo objetivo é mitigar seus problemas. [Bonomi et al. 2012] propuseram o conceito de Computação em Névoa, onde a mesma estende os recursos da computação em nuvem para a borda da rede, disponibilizados em roteadores, *access points* e outros tipos de equipamentos [Yi et al. 2015], sendo que um conjunto destes forma então uma névoa, disponibilizando recursos como computação, rede, armazenamento em uma forma virtualizada e distribuída para os sensores das aplicações IoT [Vaquero and Rodero-Merino 2014]. Em [Verma et al. 2016], os autores propuseram um arquitetura e algoritmo cujo objetivo é resolver problemas de latência, disponibilidade de recursos, largura de banda e tempo de execução em ambientes de Computação em Névoa. No modelo proposto a arquitetura possui 3 camadas, sendo elas cliente, névoa e *cloud*. Assim, os servidores na cloud e os nós da névoa interagem entre si para que, caso a Computação em Névoa não atenda os requerimentos computacionais do cliente, então, como último recurso, a requisição é encaminhada para a *cloud*. Os autores [Song et al. 2016] propuseram um $framework$ para balanceador de carga para Computação em Névoa, o qual estende o framework de processamento paralelo Apache Hadoop [Rao and Reddy 2012] que adota o modelo de programação MapReduce [Dean and Ghemawat 2008], utilizando particionamento dinâmico dos grafos. Neste, é aplicado o emprego da rede de forma flexível, que configura o sistema de acordo com a demanda do balanceador. Tal solução é composta de quatro camadas, e o balanceador é orientado a tarefas, sendo que as mesmas podem ser alocadas para um ou mais nós da Computação em Névoa. Semelhantemente em [Oueis et al. 2015], os autores, analisam a qualidade de experiência (QoE) dos usuários, propondo um balanceador de carga em Computação em Névoa, cuja proposta tem como base o uso em *offloading* das aplicações móveis dos usuários em pequenos *clusters* computacionais, utilizando redes 5G.

Diferentemente destes trabalhos, nossa proposta é a implementação do balanceador de carga que leva em consideração dois níveis de prioridade de tarefas: prioridade zero como sendo baixa, e prioridade 1 como sendo alta, alocando tarefas de baixa prioridade para a *cloud* quando os nós da névoa estiverem sobrecarregados.

3. Proposta

Na computação em nuvem, o balanceamento de carga é uma peça-chave para prover *alta disponibilidade* e *alocação correta* das tarefas nos servidores, almejando assim um aproveitamento máximo dos recursos computacionais que estão à disposição, assim como na Computação em Névoa também. A arquitetura proposta, ilustrada na Figura 1, visa prover: i) balanceamento de tarefas entre os nós da névoa; ii) alocação de tarefas de baixa prioridade para nós com menor poder computacional; e iii) respeitar o tempo limite de execução de tarefas de tempo real.

A arquitetura proposta constitui-se de 3 camadas:

I) Aplicações IoT - formada essencialmente dos sensores e atuadores de aplicações IoE (*Internet of Everything*); Nesta camada da arquitetura tem-se sensores e atuadores, sendo eles

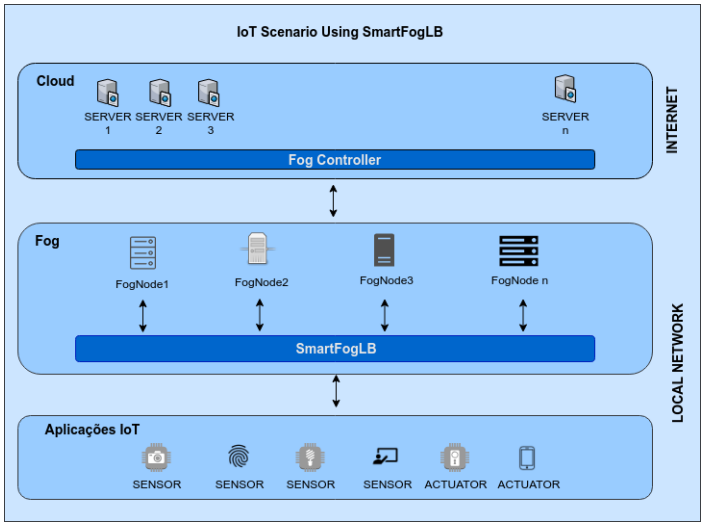

Figura 1. Modelo da Arquitetura Proposta

móveis ou não. Tais dispositivos enviam seus dados para a névoa, onde os mesmos são agregados gerando informações para tomada de decisão e após, caso seja conveniente, enviados para a *cloud*.

II) FOG (Computação em Névoa) - constituída por nós com capacidades computacionais, que recebem os dados dos sensores/atuadores IoT para processá-los; Nesta camada será introduzido o *balanceador de carga* responsável pela alocação de tarefas no nó mais conveniente, baseado nas políticas pré-definidas pelo administrador do sistema. O principal objetivo do balanceador de carga na arquitetura é de fato a distribuição das tarefas oriundas da camada inferior, alocando as mesmas dentre os nós que estão sob seu comando. Todavia, algumas tarefas possuem restrição de tempo de execução, por exemplo, uma aplicação de redes de semáforos em rodovias, o qual precisa ser realizado o mais rápido possível, para não impactar no tráfego de uma cidade inteligente.

III) Cloud (Computação em Nuvem) - é a infraestrutura da nuvem, acrescida de um servidor, o *Fog Controller*. Ele é responsável por gerir tudo o que diz respeito à Computação em Névoa e seus respectivos balanceadores de carga, a qual analisa a carga dos nós da Computação em Névoa, e em caso de sobrecarga, novos nós de processamento são instanciados na *cloud* atendendo a demanda dos usuários e mantendo tarefas de alta prioridade sempre alocadas o mais próximo do sensor. Esta camada não é objeto de estudo deste trabalho.

Também, será implementado um *daemon* que atuará em segundo plano junto ao balanceador de carga proposto, cuja função é manter uma tabela atualizada em memória dos nós disponíveis na névoa, para que o algoritmo de balanceamento de carga a consulte no momento em que a alocação de tarefas precisa ser realizada. Este, por sua vez, captura as informações de hardware do nó, bem como da carga atual do mesmo, para que o balanceador escolha o nó mais apropriado para alocar a tarefa. Assim, em caso de sobrecarga de um nó, valor superior a um $threshold_1$, ou tempo de resposta elevado, ou ainda em caso do nó não responder mais por falhas, o mesmo é retirado da tabela de nós disponíveis para o balanceamento. Também, se todos os nós estiverem com uma carga de utilização acima de um $threshold_2$, novas tarefas serão enfileiradas e posteriormente alocadas na *cloud*, salvo se as novas tarefas sejam de alta prioridade.

Isto posto, a alocação de tarefas no balanceador de carga é realizada em 3 etapas à seguir: **i)** realizado quando o sensor envia as requisições para a névoa, onde as mesmas irão chegar no balanceador de carga e não mais em um nó específico da névoa, sendo que o mesmo irá guardar em uma tabela temporária o endereço do host que fez a requisição, bem como a prioridade da mesma. O balanceador aceita tarefas com prioridades baixa (0) e alta (1), e ainda, neste passo, vai atribuir um identificador único para a tarefa; **ii)** consiste na alocação das tarefas, onde o algoritmo de balanceamento irá consumir os dados da tabela alimentada pelo *daemon*, onde os dados serão ordenados pelo menor percentual de utilização de processamento, quantidade de cores, *clock*, memória disponível, largura de banda de rede, latência do nó ao balanceador e espaço em disco, de forma decrescente; e **iii)** trata do retorno da tarefa ao balanceador, que neste trabalho não será contemplado.

4. Conclusões e trabalhos futuros

Realizar o Balanceamento de Carga tem sido uma preocupação constante em diferentes áreas da computação. Este trabalho busca a aplicação desta técnica na Computação em Névoa, uma vez que trata-se de um sistema geograficamente distribuído e virtualizado. Busca-se uma correta alocação de tarefas evitando desbalanceamento de carga e um menor tempo de execução das tarefas. Como trabalhos futuros pretende-se validar a proposta, implementando-a em simuladores ou em linguagens de programação com base de balanceamento.

Referências

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and A, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communication Surveys and Tutorials*, 17(4):2347–2376.

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*, page 13.

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Oueis, J., Strinati, E. C., and Barbarossa, S. (2015). The fog balancing: Load distribution for small cell cloud computing. *IEEE Vehicular Technology Conference*, 2015:1–6.

Rao, B. T. and Reddy, L. (2012). Survey on improved scheduling in hadoop mapreduce in cloud environments. *arXiv preprint arXiv:1207.0780*.

Song, N., Gong, C., An, X., and Zhan, Q. (2016). Fog computing dynamic load balancing mechanism based on graph repartitioning. *China Communications*, 13(3):156–164.

Stojmenovic, I. and Wen, S. (2014). The Fog Computing Paradigm: Scenarios and Security Issues. *2014 Federated Conference on Computer Science and Information Systems*, 2:1–8.

Vaquero, L. M. and Rodero-Merino, L. (2014). Finding your Way in the Fog. *ACM SIGCOMM Computer Communication Review*, 44(5):27–32.

Verma, M., Bhardwaj, N., and Yadav, A. K. (2016). Real Time Efficient Scheduling Algorithm for Load Balancing in Fog Computing Environment. *International Journal of Information Technology and Computer Science*, 8(4):1–10.

Yi, S., Li, C., and Li, Q. (2015). A Survey of Fog Computing. *Proceedings of the 2015 Workshop on Mobile Big Data - Mobidata '15*, pages 37–42.