Um Protótipo para Experimentos de Eficiência Energética em Nuvem *OpenStack*

Danilo da Silva Santos , Allan Edgard Silva Freitas

[1]Instituto Federal da Bahia (IFBA), Campus de Salvador, Bahia, Brasil

danilo.silva@ifba.edu.br, allan@ifba.edu.br

Abstract. *Infrastructure as a service cloud computing enables resources to be allocated on-demand to meet specific needs. Thus, a data center, hosting several nodes to provide the cloud, may have idle or underutilized resources consuming more energy than needed. This work proposes a tool that assists the energy management of hosts for a cloud on Openstack platform, one of the most used for IaaS clouds, allowing to turn off underutilized hosts, or to activate them in case of increased demand. This solution can be used in Energy Efficiency experiments, allowing analysis by measurement of different policies and methods.*

Resumo. *Nuvens computacionais baseadas em infra-estrutura como serviço permitem que recursos computacionais sejam alocados sob demanda de forma flexível. Desta forma, um datacenter, hospedando diversos nós para prover a nuvem, pode ter ociosidade de recursos com nós subutilizados consumindo energia elétrica desnecessariamente. Este trabalho propõe uma ferramenta que auxilia o gerenciamento energético de hosts de uma nuvem implementada na plataforma OpenStack – uma das mais utilizadas para nuvens IaaS–, permitindo desligar hosts subutilizados, ou ativá-los em caso de aumento de demanda. Esta solução pode ser utilizada em experimentos de Eficiência Energética, permitindo a análise por medição de diferentes políticas e métodos.*

1. Introdução

A necessidade de provimento de serviços para o atendimento a uma demanda dinâmica requer a possibilidade de alocação flexível de recursos, o que pode ser feito por meio de infraestrutura computacional em nuvem na forma de *Infrastructure as a Service - IaaS* ou infraestrutura como um serviço, modalidade que possibilita a disponibilização de recursos computacionais, como servidores e ativos de rede, através da *internet* [Chaisiri et al. 2012]. Atualmente, uma das plataformas mais utilizadas para prover infraestrutura como um serviço é o *OpenStack* [Gartner 2017].

Em um *cluster* computacional podem existir nós ociosos, implicando em um consumo de energia elétrica desnecessário. Consumo de energia elétrica é o principal custo operacional de *data centers* [Westphall and Villarreal 2013] e isto pode ser contornado se houver uma solução que faça uma gestão deste ambiente.

O *OpenStack* não provê uma solução que possibilite fazer gestão energética, monitorando a corrente elétrica e permitindo a mudança de estado dos *hosts* computacionais, através do desligamento de nós ociosos e acionamento de nós desligados, observada a qualidade do serviço.

A proposta é de ferramenta que monitore carga de nós da nuvem *OpenStack*, e interaja com os componentes da plataforma para desligamento e ativação dos nós, conforme a demanda de processamento e uma estratégia escolhida. Como prova de conceito, um conjunto simples e *naïve* de regras para gestão energética é exercitado. Com esta solução, experimentos para gerência de consumo de energia elétrica em nuvens *OpenStack* podem ser conduzidos.

2. Trabalhos Correlatos e Motivação

O *OpenStack Neat* [Beloglazov and Buyya 2015] é um *framework* que permite detecção de sobrecarga e subutilização, seleção e alocação dinâmica de máquinas virtuais em nuvem *OpenStack* para eficiência energética, mas esta solução não atua no desligamento de nós ociosos: nós suspensos consomem em torno de 10,4W de energia, enquanto um *host* totalmente desligado não consome energia [Meisner et al. 2009].

O trabalho de [Yang et al. 2013] apresenta uma abordagem que gerencia os recursos, permitindo o desligamento, mas que não exercita a possibilidade de religar nós caso necessário. De forma similar, em [Chen et al. 2015] há alocação dinâmica de recursos para economia de energia, mas *hosts* não são reativados caso haja necessidade.

Em [Cima et al. 2015], o *framework Kwapi* [OpenStack 2017a] obtém o consumo de energia de *hosts* e utiliza da interface inteligente de gerenciamento de energia (IPMI) e de componente de software da *IBM Active Energy Manager*. Experimentos de gerência de energia com relocação de *VMs* e desligamento de hosts foram exercitados, mas a solução depende da IMPI e do componente da IBM, sendo portanto restritiva.

Observados os trabalhos correlatos, verificamos a necessidade de uma solução escalável e que permita automatizar o desligamento e religamento dos *hosts* conforme a demanda. Nossa ferramenta não se baseia em componentes proprietários e permite acoplar diferentes algoritmos de gerência energética.

3. Arquitetura da Solução

A solução proposta, desenvolvida com a linguagem de programação *Python*, é capaz de (1) monitorar os *hosts* que provêm recursos computacionais para a plataforma *OpenStack* e (2) alterar o estado destes *hosts* através do envio de comandos para desligar ou ligar, buscando a eficiência energética e manutenção da qualidade do serviço. O código-fonte e o manual de instalação e uso estão disponíveis em https://github.com/dssantos/Cloud-Energy-Saver. Um vídeo de demonstração está acessível em https://youtu.be/JgP-1g3kOWI.

3.1. Implantação da Plataforma *OpenStack*

O *OpenStack* é uma plataforma de computação em nuvem de código aberto para criar nuvens privadas e públicas, capaz de controlar recursos de processamento, armazenamento e rede de um *datacenter*, por meio de Infraestrutura como um Serviço (*IaaS*) com uma gama de serviços complementares. Cada serviço oferece uma interface de programação de aplicativo (*API*) que facilita esta integração [OpenStack 2017b].

Alguns destes serviços são: (a) *Horizon*, portal *web* de autoatendimento para gerenciar todos os serviços; (b) *Nova*, gerência das instâncias computacionais (i.e. máquinas virtuais); (c) *Neutron*, conectividade de rede como serviço – permitindo topologias de rede

avançadas; (d) *Swift* armazenamento de objetos; (e) *Cinder*, armazenamento de blocos para uso das instâncias; (f) *Keystone*, autorização e autenticação; (g) *Glance*, manutenção de imagens de máquinas virtuais; e (h) *Ceilometer*, mensuração do uso dos recursos computacionais da nuvem.

O *OpenStack*, para prover uma infraestrutura básica, necessita de alguns *hosts* para exercer as funções de controlador, rede e computação. Os cenários de implantação podem ser bastante variados, de forma que pode existir a necessidade de utilizar *hosts* dedicados para cada uma destas funções, ou todas elas serem executadas em um mesmo *host*.

Para o uso da ferramenta, assume-se que a nuvem computacional possui um *host* que atua como controlador (*Controller*) e outros atuam com a função de computação (*Compute*), conforme a Figura 1.

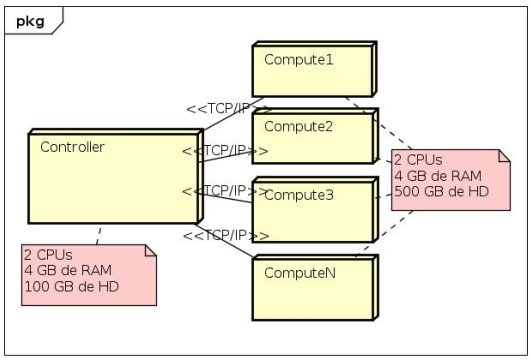

Figura 1. Diagrama de Implantação

Um uso otimizado desta nuvem sugere que a quantidade de *hosts* ativos deve obedecer a um *trade-off* entre a demanda de uso das máquinas virtuais e o consumo energético mensurado. A ferramenta proposta permite utilizar diferentes estratégias para alcançar este *trade-off*. Para o funcionamento da solução proposta, serão utilizadas funcionalidades nativas do serviço *Nova* referentes aos *hosts* utilizados. Estas informações serão coletadas por *API* e comandos nativos do sistema operacional hospedeiro. O sistema operacional hospedeiro deste ambiente sob o qual a plataforma *Openstack* executa é o Linux.

3.2. Componentes da Solução Proposta

Na Figura 2 é possível observar os componentes da solução proposta e as suas interações com componentes da plataforma *OpenStack*.

1. Registrador: é a funcionalidade que precisa ser executada inicialmente, com todos os *hosts* do ambiente do *OpenStack* em funcionamento. Tem a finalidade de registrar os *hosts* existentes no ambiente e seus respectivos endereços MAC, que serão utilizados posteriormente para ligá-lo;
2. Verificador: é a rotina que ficará executando em intervalos de tempo com o objetivo de identificar os estados dos *hosts*, obter a quantidade de *VMs* em execução e o consumo de memória RAM dos *hosts* ativos e acionar o desligamento ou inicialização de *hosts*;
3. MudaEstado: é acionado pelo componente Verificador para ligar ou desligar *hosts*.

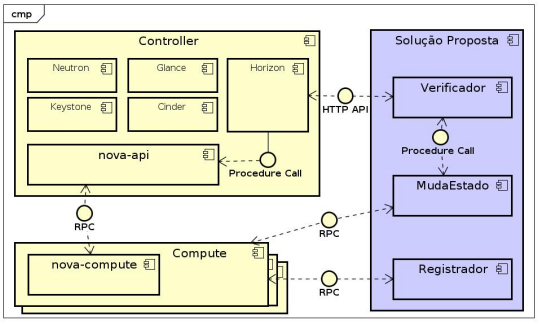

Figura 2. Diagrama de Componentes

3.3. Monitoramento e Classificação dos *Hosts*

O monitoramento dos *host* da plataforma *OpenStack*, realizado pelo componente Verificador tem o objetivo de coletar informações sobre a existência de *VMs* em execução e os recursos de *hardware* utilizados, como a memória RAM.

Com base nestas informações, conforme Figura 3, é possível classificar os *hosts* da seguinte forma:

1. Ativo: são os *hosts* que possuem pelo menos uma *VM* em execução;
2. Ocioso: são *hosts* que não possuem nenhuma *VM* em execução;
3. Desligado: são os *hosts* que foram desligados devido à ociosidade.

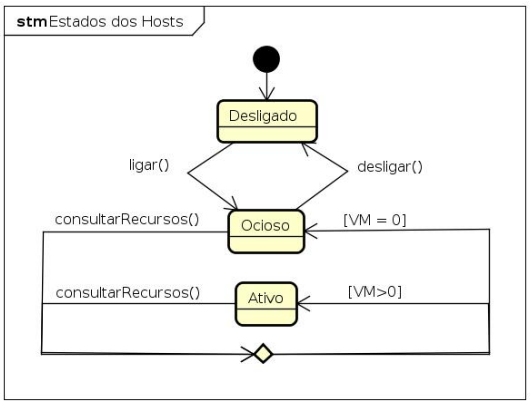

Figura 3. Diagrama de Estado dos *Hosts*

O *OpenStack* adota como limitador da alocação dos recursos virtuais as proporções de 16:1 para os núcleos de processador e de 1,5:1 para a memória RAM [OpenStack 2017c].

Por exemplo, para cada núcleo do *host* físico, podem ser instanciados 16 núcleos virtuais, ou seja, 8 máquinas virtuais com 2 núcleos. Da mesma forma que, se existirem 10 GB de memória RAM no *host* físico, poderão ser alocados 15 GB de memória RAM para as máquinas virtuais hospedadas nele.

Devido a esta alavancagem do uso de processador, quantidade de instâncias e processadores pode não ser uma métrica adequada para aferir carga de sistema. Propomos que esta carga do sistema seja estimada pela média do percentual de memória RAM utilizada pelos *hosts* ativos.

3.4. Alteração do Estado dos *Hosts*

Com a ociosidade de alguns *hosts*, não há necessidade destes permanecerem ligados, assim, a solução realiza o desligamento destes *hosts* ociosos.

Sempre que o componente Verificador identifica a ociosidade de *hosts*, o componente MudaEstado é acionado para executar o desligamento através do comando *shutdown now*. Por outro lado, caso seja identificada uma sobrecarga, outros *hosts* podem ser acionados, utilizando *Magic Packets*, conforme o padrão *Wake On Lan*[Moreira 2017]. O *Shell Script Etherwake* [Mkssoftware 2018] foi utilizado para este fim.

A decisão pelo desligamento depende da qualidade de serviço exigida, tendo em vista que, em determinado momento podem surgir demandas por recursos computacionais e estes *hosts* precisam ser ligados.

O fluxo de atividades que a solução proposta executa está descrito a seguir:

1. Inicialmente, com todos os *hosts* ligados, a aplicação executa o componente Registrador para descobrir os que estão executando o serviço *nova-compute* do *OpenStack* e armazena o endereço MAC destes.
2. A solução realiza uma verificação para obter a quantidade de *VMs* e memória RAM em uso, em seguida classifica cada *host*(*compute*) como ativo, ocioso ou desligado.
3. A solução calcula a carga do sistema a partir da informação dos *hosts* ativos e toma uma decisão a depender do valor obtido e a quantidade de *hosts* ociosos.
4. De acordo com a estratégia implantada, a solução pode iniciar *hosts* que estejam desligados ou desligar *hosts* ociosos, reconfigurando o sistema.

A inicialização de um *host* e a sua identificação pode ser demorada, e a depender do *hardware* e das configurações do *host*. Em experimentos preliminares mensuramos um tempo de ativação de até 90 segundos. Considerando isto, a estratégia adotada deve permitir um nível de ociosidade de folga de modo a evitar *thrashing* com ativações e desativações em sequencia por operar nos limites.

Uma estratégia *naïve* é experimentada na Prova de Conceito: definimos níveis de serviços a partir de patamares máximo e médio. Por exemplo, o sistema terá uma melhor qualidade de serviço se for definido um patamar máximo de utilização de memória RAM de 40%. Por outro lado, caso este patamar esteja em 80%, o sistema priorizará a eficiência energética, e somente inicializará novos *hosts* caso o sistema ultrapasse este limite.

4. Prova de Conceito e Demostração

O experimento foi realizado em um ambiente composto por microcomputadores, devidamente configurados com a plataforma *OpenStack*. Os *hosts* que executavam o serviço *nova-compute* foram os alvos da medição, pois estes eram desligados e ligados.

A corrente elétrica foi medida com um sensor ACS712 [MicroSystems 2018] conectado a um microcontrolador NodeMCU ESP8266 [Systems 2018] com *Wireless* integrado que capturou as informações da corrente de um circuito elétrico no qual apenas os *hosts computes* estavam conectados.

Para gerar carga computacional no ambiente do experimento foi utilizada uma aplicação com características similares ao CloudGI [Nascimento et al. 2015], um geren-

ciador de réplicas para o ambiente de computação em nuvem *OpenStack*, que permite inicializar instâncias de forma automatizada.

Desta forma, inicializou-se uma instância de cada vez, em intervalos de 30 segundos, até alcançar a quantidade de 50 *VMs* em execução. Depois disto, após 30 segundos, as instâncias foram desligadas uma a uma, no mesmo intervalo de tempo da inicialização.

O experimento foi executado em dois momentos distintos, que posteriormente foram comparados: (1) sem utilizar a solução proposta, durou 60 minutos e resultou uma corrente média de 2,11 Amperes, conforme Figura 4. Em seguida, (2) com a solução em funcionamento, a execução durou 60 minutos e apresentou uma corrente média de 1,78 Amperes, conforme Figura 5.

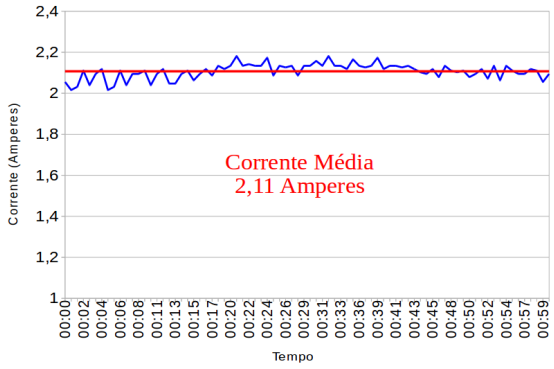

Figura 4. Corrente média sem o uso da solução proposta

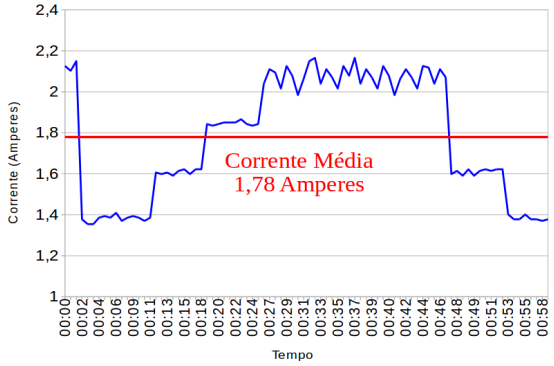

Figura 5. Corrente média com o uso da solução proposta

Ao observar a evolução do gráfico apresentado na Figura 5 percebe-se os seguintes marcos:

- 00:00 - a verificação é iniciada. Existem 4 *hosts* ligados;
- 00:01 - o Verificador identifica a ociosidade do *cluster* e desliga 3 *hosts*;
- 00:11 - a carga do *host* ativo atinge o patamar máximo, então a solução inicializa 1 *host*, totalizando 2 ligados;
- 00:18 - a carga média dos *hosts* ativos atinge o patamar máximo, então a solução inicializa mais 1 *host*, totalizando 3 ligados;
- 00:24 - a carga média dos *hosts* ativos atinge o patamar máximo novamente, e a solução inicializa mais outro *host*, totalizando 4 ligados;

- 00:46 - a carga média dos *hosts* ativos reduz abaixo do patamar médio, então a solução desliga os 2 *hosts* ociosos, mantendo 2 ligados;
- 00:51 - a carga do *host* ativo reduz abaixo do patamar médio, e a solução desliga mais outro *host* que ficou ocioso, mantendo apenas 1 ligado.

Durante o período de execução, com a solução em funcionamento, foi adotado um patamar médio de 30% e máximo de 40% de uso de memória RAM do sistema como critério para inicialização e desligamento de *hosts*, visando garantir a qualidade do serviço. No entanto, estes patamares podem ser alterados para priorizar a redução da corrente elétrica.

Com o resultado, observou-se que a corrente elétrica neste cenário foi reduzida em 15,6% ao utilizar a solução proposta. Diante disto, fica demonstrada que a utilização de uma solução de gerenciamento de *hosts*, de um ambiente de computação em nuvem que utiliza a plataforma *OpenStack*, oferece uma redução de corrente elétrica.

A demonstração proposta para o SBRC 2019 é de uma reprodução da prova de conceito, utilizando 4 nós (1 como controlador e 3 como *hosts*), um pequeno *switch* de rede, um filtro de linha, um sensor ACS712 [MicroSystems 2018] conectado a microcontrolador NodeMCU ESP8266 [Systems 2018] e um monitor apresentando a execução da ferramenta e ações de monitoramento e atuação nas instâncias da nuvem.

5. Considerações Finais

O presente trabalho propõe uma ferramenta que permite monitoramento da corrente elétrica e desligamento e religamento de *hosts*, reconfigurando o ambiente *OpenStack* e permitindo uma otimização que implica na redução do consumo de energia da nuvem computacional.

Algumas lacunas identificadas em trabalhos relacionados foram atendidas: (1) possibilidade de utilização de *hosts* computacionais diversos, como microcomputadores ou servidores, (2) possibilidade de equilíbrio entre a redução da corrente e a qualidade do serviço, (3) escalabilidade da solução e (4) medição da corrente do circuito elétrico dos *hosts*.

Foi exercitada uma Prova de Conceito baseado em um conjunto sumário de regras que definem ações de ativação ou desativação de acordo com o percentual de uso de memória RAM dos nós. Deve-se ressaltar que este algoritmo pode ser facilmente substituído por outras abordagens, permitindo diferentes mecanismos para obter o *trade-off* adequado entre consumo de corrente e a qualidade do serviço.

Desta forma, propomos que esta ferramenta possa ser utilizada como base para experimentos de eficiência energética em nuvens *OpenStack*.

Referências

Beloglazov, A. and Buyya, R. (2015). Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds. *Concurrency and Computation: Practice and Experience*, 27(5):1310–1333.

Chaisiri, S., Lee, B.-S., and Niyato, D. (2012). Optimization of resource provisioning cost in cloud computing. *IEEE Transactions on Services Computing*, 5(2):164–177.

Chen, C.-C., Sun, P.-L., Yang, C.-T., Liu, J.-C., Chen, S.-T., and Wan, Z.-Y. (2015). Implementation of a cloud energy saving system with virtual machine dynamic resource allocation method based on openstack. In *Parallel Architectures, Algorithms and Programming (PAAP), 2015 Seventh International Symposium on*, pages 190–196. IEEE.

Cima, V., Grazioli, B., Murphy, S., and Bohnert, T. M. (2015). Adding energy efficiency to openstack. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2015*, pages 1–8. IEEE.

Gartner (2017). Using openstack for private cloud. `https://www.gartner.com/doc/3459617/using-openstack-private-cloud`. Acessado em 22/08/2017.

Meisner, D., Gold, B. T., and Wenisch, T. F. (2009). Powernap: eliminating server idle power. *ACM Sigplan Notices*, 44(3):205–216.

MicroSystems, A. (2018). Acs712: Fully integrated, hall-effect-based linear current sensor ic with 2.1 kvrms voltage isolation and a low-resistance current conductor. `https://www.allegromicro.com/en/Products/Current-Sensor-ICs/Zero-To-Fifty-Amp-Integrated-Conductor-Sensor-ICs/ACS712.aspx`. Acessado em 29/09/2018.

Mkssoftware (2018). etherwake - send a wake-on-lan magic packet. `https://www.mkssoftware.com/docs/man1/etherwake.1.asp`. Acessado em 29/09/2018.

Moreira, A. (2017). Wake on lan. `http://www.dei.isep.ipp.pt/~andre/documentos/wol.html`. Acessado em 08/10/2017.

Nascimento, P. S., Freitas, and Silva, A. E. (2015). Cloudgi, gerenciando instâncias de um serviço replicado em uma plataforma de computação em nuvem. In *Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (Salão de Ferramentas)*.

OpenStack (2017a). Kwapi's developer documentation. `https://kwapi.readthedocs.io/en/latest/`. Acessado em 22/08/2017.

OpenStack (2017b). Open source software for creating private and public clouds. `https://www.openstack.org/`. Acessado em 22/08/2017.

OpenStack (2017c). Overcommitting cpu and ram. `https://docs.openstack.org/arch-design/design-compute/design-compute-overcommit.html`. Acessado em 03/10/2017.

Systems, E. (2018). Esp8266: Low-power, highly-integrated wi-fi solution. `https://www.espressif.com/en/products/hardware/esp8266ex/overview`. Acessado em 29/09/2018.

Westphall, C. B. and Villarreal, S. R. (2013). Princípios e tendências em green cloud computing. *Revista Eletrônica de Sistemas de Informação*, 12(1):1–19.

Yang, C.-T., Huang, K.-L., Liu, J.-C., Su, Y.-W., and Chu, W. C.-C. (2013). Implementation of a power saving method for virtual machine management in cloud. In *Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on*, pages 283–290. IEEE.