Universidade Federal do Pampa
Raul Dias Leiria

Monitoramento Energético em Nuvens Computacionais

Alegrete
2016

Raul Dias Leiria

Monitoramento Energético em Nuvens Computacionais

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Claudio Schepke

Coorientador: Aline Vieira de Mello

Alegrete

2016

Raul Dias Leiria

Monitoramento Energético em Nuvens Computacionais

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho de Conclusão de Curso defendido e aprovado em ..... de ..... de .....

Banca examinadora:

Prof. Dr. Claudio Schepke
Orientador
UNIPAMPA

Profª. Drª. Aline Vieira de Mello
Co-orientadora
UNIPAMPA

Prof. Dr. Diego Luis Kreutz
UNIPAMPA

Prof. Dr. Dalvan Jair Griebler
PUCRS

Este trabalho é dedicado à Deus e a todas as pessoas que de alguma maneira me incentivaram a continuar na caminhada do saber.

Agradecimentos

Agradeço à Deus, a meus pais Edson Leiria e Solange Leiria, a meu avô Juvenal Dias, a meus orientadores Claudio Schepke e Aline Mello, a meus colegas e amigos. Sem o apoio incondicional de vocês este trabalho não teria sido possível.

"A fé robusta dá a perseverança, a energia e os recursos que fazem vencer os obstáculos, tanto nas pequenas quanto nas grandes coisas."
(O E.S.E., cap. XIX, item 2)

Resumo

As nuvens computacionais consomem grandes quantidades de energia elétrica, sendo responsáveis por causar a emissão de 2% do dióxido de carbono mundial. Alguns gerenciadores de nuvem, como o OpenNebula, não dispõem de recursos para monitorar o consumo energético de seus serviços e infraestruturas, deixando seus administradores desinformados sobre a quantidade de energia elétrica consumida. Em razão disso, o objetivo deste trabalho é criar um sistema para monitorar o consumo energético em *data centers* virtualizados com *Kernel-based Virtual Machine* e geridos pelo OpenNebula. O sistema proposto denomina-se Monitor Energético (ME) e divide-se em Monitor Energético em Bash (MEB) para coleta dos consumos energéticos das máquinas virtuais e Monitor Energético em Web (MEW) para exibir os dados energéticos na interface Web do OpenNebula. Experimentos realizados com a ferramenta Sysbench comprovaram o funcionamento do sistema proposto durante o estresse proposital do ambiente analisado.

Palavras-chave: Virtualização. Redes de computadores. Eficiência energética. Linux. Monitoramento energético. OpenNebula.

Abstract

Computational clouds consume lots of electrical energy and are responsible for causing the issuance of 2% of globally carbon dioxide. Some cloud managers, such as OpenNebula, have no resources to monitor the energy consumption of their services and infrastructure, making the cloud administrator uninformed about the amount of electrical energy spent. Therefore, this work aims to create a system for monitoring energy consumption in data centers virtualized with Kernel-based Virtual Machine and managed by OpenNebula. The proposed system is called Monitor Energético (ME) and is divided in Monitor Energético em Bash (MEB) to collect virtual machines' energy consumptions and Monitor Energético em Web (MEW) for display energy data at OpenNebula Web interface. Experiments performed with Sysbench tool proved the correctness of the proposed system during the purposeful stress of the analyzed environment.

Key-words: Virtualization. Computer Networks. Efficiency energy. Linux. Energy monitoring. OpenNebula.

Lista de ilustrações

Lista de tabelas

Lista de siglas

BIOS Basic Input and Output System

CPU Unidade de Processamento Central

GUI Graphical User Interface

IaaS Infrastructure as a Service

IP Internet Protocol

iSCSI Internet Small Computer System Interface

KVM Kernel-based Virtual Machine

MAC Media Access Control

ME Monitor Energético

MEB Monitor Energético em Bash

MEW Monitor Energético em Web

NFS Network File System

PaaS Plataform as a Service

PID Process Identifier

QEMU Quick Emulator

RAID Redundant Array of Independent Disks

RAM Random Access Memory

RPC Remote Procedure Call

SaaS Software as a Service

SO Sistema Operacional

SQL Structured Query Language

SSH Secure Shell

TLB Translation Lookaside Buffer

USB Universal Serial Bus

VM Máquina Virtual

VMM Virtual Machine Monitor

Sumário

1 Introdução

Nos últimos anos a aderência em massa à Internet possibilitou grandes oportunidades de negócios às empresas. Com esse acontecimento a maioria das organizações de alguma maneira adentrou à Rede Mundial e precisou de infraestruturas computacionais para hospedar seus serviços de forma *online*. Nem todas as empresas estavam preparadas para isso e o aluguel de infraestruturas computacionais foi e ainda é uma possível solução.

A computação em nuvem visa o acesso, a hospedagem e gerência remota de infraestruturas e serviços independente da localidade geográfica dos envolvidos. Sua popularidade é consequência da boa consolidação de seus modelos de serviço (IaaS, PaaS, SaaS) e de seus modelos de implantação (nuvens públicas, híbridas, comunitárias e privadas) (VOGEL et al., 2016). As nuvens computacionais em sua maioria são compostas por sistemas virtualizados, o que facilita a manutenção, migração, expansão e níveis de abstração diferenciados para quem as utiliza.

Há três possíveis perspectivas para a forma de interação com as nuvens computacionais, sendo a primeira a do usuário que acessa o recurso *online*, a segunda da empresa que contrata ou é dona da nuvem, e a terceira do administrador do *data center* (nuvem). Das diversas atribuições do administrador, destaca-se para este trabalho a de monitorar os recursos computacionais (ROVEDAL et al., 2015).

O consumo energético dos *data centers* é responsável por causar a emissão de 2% do dióxido de carbono mundial (SCHULZ, 2009). A virtualização é um recurso de *software* que permite executar computadores virtuais em computadores físicos, e por essa característica também contribui na quantidade de energia elétrica consumida pelo *data center*.

Enquanto há *hardware* específico para monitorar o consumo energético de computadores físicos, não há nenhum recurso para monitorar o de computadores virtuais (máquinas virtuais) (BOURDON et al., 2011). Uma solução para isso é utilizar modelos energéticos (matemáticos) que estimem o consumo energético das máquinas virtuais através de dados coletados do kernel do sistema operacional.

Das pesquisas realizadas anteriormente ao início deste trabalho, não encontrou-se na literatura trabalhos que abordassem formas de monitorar o consumo energético de máquinas virtuais administradas pelo gerenciador de nuvem OpenNebula. Em vista disso, o objetivo deste trabalho é desenvolver um *addon* (estensor de funcionalidades) que supra essa necessidade e exiba os respectivos dados energéticos na interface *Web* do OpenNebula *Sunstone* versão 4.14.2.

O trabalho a partir da introdução, está organizado em capítulos da seguinte maneira: No capítulo 2 é apresentada a contextualização dos tópicos mais pertinentes ao leitor. Os trabalhos relacionados encontram-se no capítulo 3, no capítulo 4 está a metodologia e o capítulo 5 contém o desenvolvimento e a validação. Por fim, o capítulo 6 apresenta a conclusão.

2 Contextualização

2.1 Consumo Energético

Utilizar energias renováveis, bem como poupar energia elétrica são grandes contribuições para o planeta. Os *data centers* são vilões para o meio ambiente, em razão do seu alto consumo de energia elétrica, criando a necessidade de ferramentas de *hardware* ou *software* que estimem o consumo energético em diferentes granularidades.

As medições via *hardware* interceptam o consumo energético do servidor com ferramentas como a PowerSpy2 (ALCIOM, 2015) e o transmitem via meios guiados. O processador também possui recursos físicos específicos para medir o quanto o seu circuito está consumindo de energia elétrica, utilizando para isso registradores que estão presentes na maioria das arquiteturas computacionais modernas.

Quando não há *hardware* disponível para medir o consumo energético, são feitas estimativas via *software* com base em estatíticas de uso dos recursos do sistema, como as do processador, rede, memória e etc. A coleta dessas estatísticas no nível mais próximo do *hardware* geralmente é feita pelos contadores de desempenho do kernel.

2.2 Sistema Operacional Linux

A diversidade de marcas e modelos de *hardware* é enorme. Uma mínima variação num determinado circuito pode dar a ele outro nome e funcionalidades, criando assim um verdadeiro *mix* de caracaterísticas e peculiaridades. O sistema operacional existe justamente para lidar com a heterogeneidade dos *hardwares*, sendo uma de suas principais funções abstrair as diferenças dos dispositivos físicos numa interface uniforme para as aplicações (TANENBAUM, 2010).

O shell e a Graphical User Interface (GUI) possibilitam ao usuário, por meio dos dispositivos de entrada e saída, a interação com o sistema operacional e seus aplicativos. No Linux, sistema operacional *open source* e *free*, o shell possui uma maior gama de recursos para a interação do usuário. Já o Windows, também possui um shell, porém, muitas ações realizadas pelo Windows Explorer não podem ser reproduzidas no seu interpretador de comandos Prompt de Comando. Em razão disso, e pela característica de que o Linux não possui custos ou restrições na sua utilização, sua adoção em *data centers* é bastante comum.

O GNU/Linux, ou simplesmente Linux, é um sistema baseado nos princípios do UNIX. Seu kernel é distintamente separado do sistema operacional, o que facilita a mo-

dularização de código fonte. Sua estrutura de diretórios possui formato hierárquico em árvore, e geralmente são adotados os sistemas de arquivos ext3, ext4 e ReiserFS em caso de sistemas isolados, ao passo que para sistemas distribuidos, pode ser utilizado o Network File System (NFS).

2.3 Virtualização

A demanda exponencial de acesso aos serviços da Internet impõe aos provedores o uso de *data centers* sofisticados, com baixo custo e com bom índice de eficiência energética. A virtualização é um recurso que veio ao encontro dessas duas exigências, pois através dela é possível implementar computadores virtuais (máquinas virtuais) equivalentes a computadores físicos. Isso economiza espaço físico e energia, facilita a manutenção, e aumenta a disponibilidade sempre que há replicações dos serviços. Tanenbaum (2010) mostra que há duas abordagens para a virtualização, sendo respectivamente hipervisor tipo 1 e tipo 2.

No hipervisor tipo 1, representado pela Figura 1, e também chamado de Virtual Machine Monitor (VMM), cada máquina virtual é representada por um Process Identifier (PID) no sistema operacional hospedeiro, e o VMM é o próprio sistema operacional. Sua tarefa, em relação as máquinas virtuais, é gerenciar múltiplas instâncias do *hardware* e agir como uma interface entre os sistemas operacionais hóspedes e o *hardware*. Apesar dos sistemas hóspedes acreditarem estar no modo núcleo, as Máquinas Virtuais (VMs) não fazem acesso direto ao *hardware*, cabendo ao hipervisor ser acionado para isso. Caso a tecnologia de virtualização da Intel ou da AMD não estiver presente na Unidade de Processamento Central (CPU), e houver alguma instrução que necessite realizar uma operação de entrada e saída, essa instrução sensível não poderá ser capturada pela armadilha de instruções privilegiadas, e por consequência as máquinas virtuais não poderão ser inicializadas.

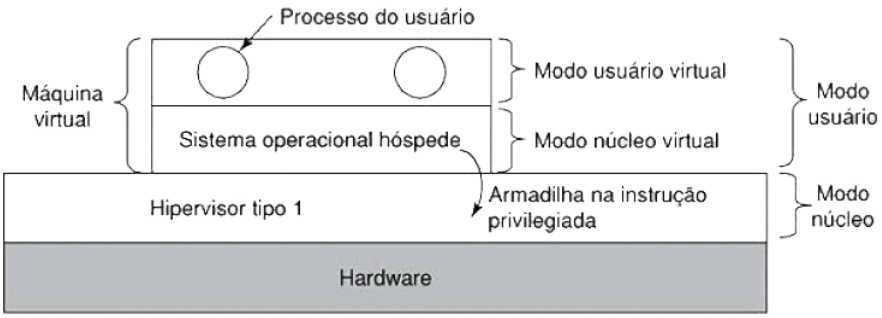

Figura 1 – Hipervisor tipo 1

Fonte: (TANENBAUM, 2010)

O hipervisor tipo 2 difere-se do tipo 1 na característica de não precisar de tecno-

logias de virtualização no processador. Por isso, esse hipervisor atua como um processo do sistema operacional hospedeiro, que recebe e emula instruções de entrada e saída dos sistemas hóspedes, as executa e retorna os resultados para os sistemas operacionais solicitantes. Esse processo é denominado Tradução Binária e ocorre sempre que instruções sensíveis precisam ser processadas.

Tanenbaum (2010) afirma que o hipervisor tipo 1 não necessariamente é superior em termos de desempenho ao tipo 2. Por o tipo 1 possuir as armadilhas para captura de instruções advindas dos sistemas hóspedes, as memórias *caches*, o Translation Lookaside Buffer (TLB) e a tabela de previsão de desvios tornam-se ineficazes, criando um maior custo de acesso ao *hardware*.

Na Paravirtualização, representada pela Figura 2, ou também chamada de Virtualização Assistida por Sistema Operacional, o hipervisor é denominado microkernel e é necessário que os sistemas operacionais hóspedes sofram modificações internas, a fim de que substituam as instruções sensíveis por chamadas de rotinas do hipervisor, dispensando assim o uso da Tradução Binária (TANENBAUM, 2010).

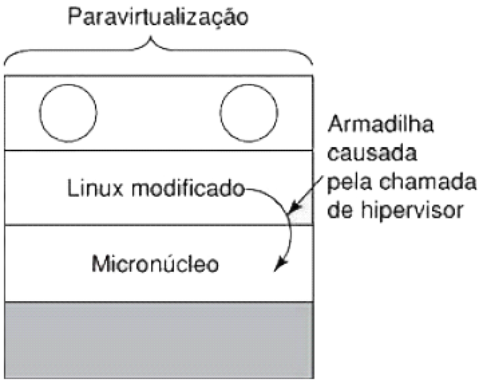

Figura 2 – Paravirtualização

Fonte: Adaptação de (TANENBAUM, 2010)

2.4 Kernel-based Virtual Machine

O Kernel-based Virtual Machine (KVM) é um módulo do kernel do Linux que implementa um VMM (JANG, 2010). Ele está presente na maioria das distribuições Linux, é *open source* e graças as tecnologias de virtualização inclusas em CPUs da Intel e da AMD, o KVM pode executar as máquinas virtuais diretamente sobre o *hardware* (PETERSEN, 2010).

Do ponto de vista do sistema operacional, cada VM do KVM é tratada como um processo, e a interação do VMM com o processador se dá por meio de módulos do kernel (kvm-intel e kvm-amd). No espaço do usuário, o KVM é visto como uma interface para os processos interagirem com a virtualização, sendo necessária uma integração com

o Quick Emulator (QEMU) para emulação dos dispositivos de entrada e saída, ou com o VirtIO para paravirtualização (o Sistema Operacional (SO) hóspede não vê o hardware diretamente e sabe que está sendo virtualizado (TORALDO, 2012)). A Figura 3 ilustra o funcionamento e a interação entre o QEMU e o KVM.

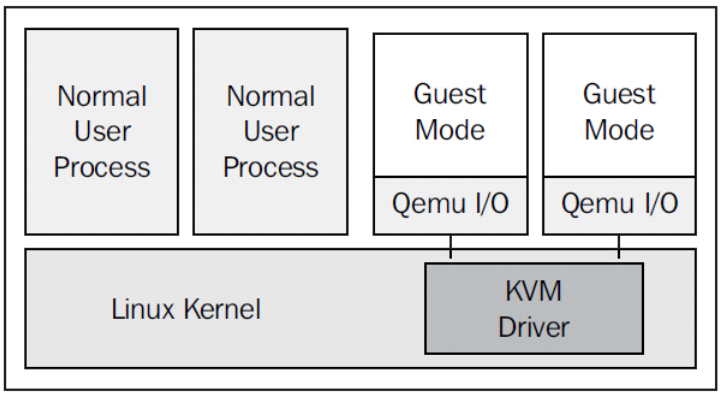

Figura 3 – Interação entre o QEMU e o KVM

Fonte: (TORALDO, 2012)

2.5 Computação em Nuvem

A computação em nuvem compreende uma infraestrutura computacional altamente disponível e confiável, que oferece serviços *online* e em larga escala. O usuário locatário pode expandir o *hardware* contratado sob demanda, sem paralisar os serviços que estejam em funcionamento. Segundo (NETO, 2015), há quatro principais modelos de implantação para a computação em nuvem:

- **Nuvem Privada**: É uma infraestrutura que geralmente é hospedada e administrada pela própria empresa que vai utilizar os serviços. A vantagem desse tipo de implantação é o maior nível de segurança, já que os dados não serão hospedados por terceiros.

- **Nuvem Pública**: Segue quase a mesma ideia da nuvem privada, porém, nesse modelo a infraestrutura é hospedada e gerenciada por provedores terceirizados. O nível de segurança tende a ser menor, pois, há outras empresas que também estão hospedadas pelo mesmo provedor, e caso alguma delas sofra um ataque cibernético, toda a segurança pode ser comprometida.

- **Nuvem Comunitária**: Nesse modelo, ao invés de uma única empresa acessar e gerenciar a nuvem, há um grupo de entidades que compartilham os mesmos interesses em uma determinada área, e por conseguinte, também os mesmos serviços.

- **Nuvem Híbrida**: Composta por dois ou mais modelos de implantação. Objetiva interligar as diferentes nuvens para o compartilhamento de recursos entre elas.

Segundo (JAMSA, 2012), os modelos de implantação especificam como os recursos da nuvem são compartilhados, ao passo que, os modelos de serviço descrevem as maneiras utilizadas para interação entre usuário e nuvem, conforme explanado e enumerado a seguir:

- Software as a Service (SaaS): É uma aplicação (programa) que possui interface gráfica e está disponível para acesso *online*. Segue as restrições dos modelos de implantação.
- Plataform as a Service (PaaS): Consiste em uma plataforma composta por ferramentas de desenvolvimento, banco de dados (VOGEL et al., 2015) e etc, que fornece aos programadores um ambiente para testes e implantações de suas aplicações.
- Infrastructure as a Service (IaaS): É uma *pool* de recursos para abstrair a complexidade computacional da infraestrutura (*hardware* e virtualização) (ROVEDAL; VOGEL; GRIEBLER, 2015), onde os administradores que locam o serviço podem criar e gerenciar seus próprios ambientes.

As nuvens computacionais possuem diferentes níveis de abstração, o que torna necessário ferramentas de gerenciamento de nuvem, bem como o OpenStack e o OpenNebula, para diminuir a complexidade das tarefas de administração.

2.5.1 OpenStack

O OpenStack é um gerenciador para nuvens IaaS públicas e privadas, projetado para ser escalável e prático (PEPPLE, 2011). Seu funcionamento está associado a três componentes, conforme a Figura 4.

Figura 4 – Componentes do OpenStack

Fonte: (PEPPLE, 2011)

O componente *OpenStack Compute* é representado pelo *software* 'Nova' e permite gerenciar as máquinas virtuais. Para gerenciar a redundância dos *storages*, o componente *OpenStack Object Store* utiliza o software *Swift*. Já o software *Glance*, representado pelo componente *OpenStack Image Service*, provê serviços de busca e armazenamento para discos virtuais.

2.5.2 OpenNebula

O OpenNebula (PROJECT, 2015), que é um gerenciador de nuvem para IaaS, suporta núvens privadas, públicas e híbridas (TORALDO, 2012). A sua estruturação está dividida em três níveis, conforme mostra a Figura 5 .

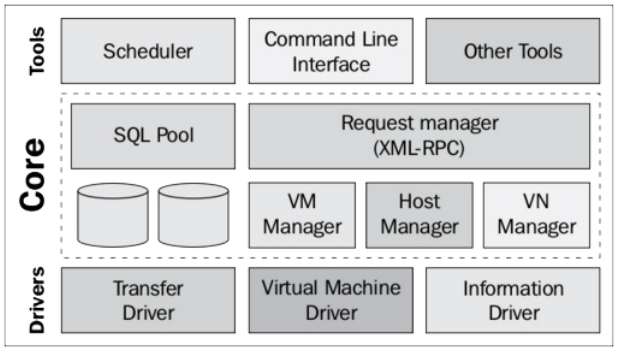

Figura 5 – Estrutura do OpenNebula

Fonte: (TORALDO, 2012)

O nível *drivers* possui componentes que se comunicam diretamente com o sistema operacional. O *Transfer Driver* é responsável por gerenciar os discos virtuais das VMs nos *storages* (clonar, deletar, alterar, ...), sendo compatível com NFS, Internet Small Computer System Interface (iSCSI), ou transferências via Secure Shell (SSH). O componente *Virtual Machine Driver* está encumbido de gerenciar o ciclo de vida das máquinas virtuais no hospedeiro (*deploy*, *shutdown*, migração, ...), e em razão disso, possui comunicação com o hipervisor. O *Information Driver* obtém o estado atual das instâncias das VMs e dos hospedeiros (coleta informações sobre eles), ele trabalha de forma específica com cada hipervisor, e troca informações via SSH.

O nível *core* é um nível intermediário, que cria uma abstração entre o nível *drivers* e o *tools*. Ele está dividido em cinco componentes, sendo o *Request Manager* para prover uma interface XML-RPC que gerencie e obtenha informações acerca das entidades (*hosts*, máquinas virtuais e rede virtuais); O *SQL Pool* para armazenar informações sobre as entidades; O *VM Manager* para cuidar do ciclo de vida das VMs; O *Host Manager* para obter informações sobre os hospedeiros e a forma de interagir com eles; E por fim, o *VN Manager*, que é responsável por gerar endereços Media Access Control (MAC) e Internet Protocol (IP).

O nível *tools* é o mais alto da estrutura e possui basicamente dois componentes. O *scheduler* busca por hospedeiros para implementar máquinas virtuais, enquanto a interface de linha de comando serve para gerenciar as entidades.

2.5.3 Análise de Desempenho dos Gerenciadores de Nuvem OpenNebula e OpenStack

Na análise realizada por (MARON et al., 2014), é feito um comparativo de desempenho entre as ferramentas *open source* OpenNebula e OpenStack. Por meio de testes utilizando *benchmarks*, os autores concluiram que dependendo dos aspectos considerados na avaliação, cada ferramenta pode ou não se sobresair perante a outra. Nos casos onde as aplicações utilizaram excessivamente o disco, o OpenStack obteve melhor desempenho, já o OpenNebula apresentou melhor performance com processos que façam maior uso de rede, processador e memória.

3 Trabalhos Relacionados

A virtualização veio para diminuir o consumo exacerbado de recursos físicos nos *data centers*, mas trouxe consigo a falta de meios físicos embarcados para medir o consumo energético de máquinas virtuais. Em razão disso, nesse capitulo estão descritos trabalhos científicos que contém as ferramentas e técnicas de *software* mais relevantes para estimar o consumo energético de *data centers* (nuvem). Os trabalhos que aqui constam, num primeiro momento, foram encontrados na biblioteca digital Google Scholar, utilizando as seguintes palavras-chave: *data center*, *power*, *virtual machine*, *energy*, *cloud*, *metric*, *Kernel-based Virtual Machine*, *OpenNebula*. Num segundo momento, propositando fazer um refinamento na escolha dos trabalhos em relação às técnicas ou ferramentas, foram priorizados os seguintes critérios:

1. Realizar de forma especifica medições em máquinas virtuais, processos ou nuvem.
2. Independer de arquitetura computacional.
3. Ser *open source* e estar disponível para *download*.
4. Possuir compatibilidade com o KVM.
5. Haver praticidade na instalação e execução, de modo que haja independência de *softwares* extras nos sistemas operacionais hóspedes.
6. Possuir uma maneira de totalizar os consumos energéticos dos hospedeiros, VMs e *data center*.
7. Possuir integração com alguma ferramentade gerenciamento de nuvem, preferencialmente o OpenNebula (PROJECT, 2015).
8. Disponibilizar uma forma de exportar os dados estimados.

Apesar do esforço para selecionar trabalhos específicos à temática dessa monografia, nem sempre foi possível encontrar em todos as especificidades demarcadas para a pesquisa bibliográfica. Por isso, a Tabela 1 apresenta os trabalhos selecionados e indica os critérios atendidos pelos mesmo, onde o '+' representa atendimento ao critério, o 'NA' não atendimento e o 'NC' nada consta sobre o mesmo. O resumo de cada trabalho é apresentado a seguir.

Bohra e Chaudhary (2010) propõe uma ferramenta denominada VMeter, capaz de medir o consumo energético de máquinas virtuais do hipervisor Xen. É necessário que o sistema operacional hospedeiro possua em seu kernel o módulo que suporte esse

hipervisor. As medidas energéticas são obtidas por meio do monitoramento de recursos como CPU, cache, disco e Random Access Memory (RAM), via *performance counters* e via *software* de monitoramento de disco nativo do Linux, o iostat. O total energético de cada VM é obtido relacionando às medidas obtidas com o total de energia consumida pelo *host*. Atualmente essa ferramenta não possui integração com gerenciadores de nuvem.

Tabela 1 – Comparativo dos Trabalhos Relacionados

Trabalho	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
(BORGETTO; STOLF, 2014)	+	NC	NC	+	NC	NC	+	NC
(LI et al., 2012)	+	+	NC	NA	NC	NC	NC	NC
(BERTRAN et al., 2011)	+	+	NC	+	NC	NC	NC	NC
(ZHAI et al., 2014)	+	NA	NC	NC	NC	NA	NC	NC
(MARCU; TUDOR; FUICU, 2011)	+	+	NC	NA	NC	NC	NC	NC
(COLMANT et al., 2015)	+	+	+	+	+	NA	NA	+
(VERSICK; WABMANN; TAVANGARIAN, 2013)	+	+	NC	NA	NC	+	NA	NC
(CHENGJIAN et al., 2013)	+	NC	NC	NA	NC	NC	NC	NC
(KANSAL et al., 2010)	+	NC	NC	NA	NC	NC	NC	NC
(BOHRA; CHAUDHARY, 2010)	+	NC	NC	NA	NC	NC	NA	NC
(BOURDON et al., 2011)	+	+	+	+	+	NA	NA	+

Kansal et al. (2010) apresenta uma ferramenta genérica para sistemas operacionais hospedeiros Microsoft Windows. A dita ferramenta denomina-se Joulemeter e possibilita medir o consumo energético de processos e máquinas virtuais. A obtenção dos dados de consumo de energia ocorre por meio de modelos energéticos, que utilizam como parâmetro as estatísticas de uso dos recursos do sistema; Cada consumo pode ser atribuído a sua respectiva VM ou a um processo. A Joulemeter está disponível para download, independe de *hardware* e não é *open source*.

Versick, Wabmann e Tavangarian (2013) propõe uma técnica que mede o consumo energético de máquinas virtuais e que se aplica a sistemas operacionais Linux, contanto que esses suportem em seu kernel o hipervisor Xen. Os consumos de energia são coletados por meio do medidor CLM5-IP. Esse aparelho intercepta o consumo energético de dispositivos que estejam ligados a ele, e envia-os via rede de computadores. Os dados energéticos, junto aos dados numéricos relativos a utilização do servidor *host*, são utilizados num modelo de energia autoadaptativo, que gera como resultado os consumos energéticos das VMs. Essa solução independe de *hardware* e não possui integração com gerenciadores de nuvem, mas, por meio de um *web service* é possível receber, centralizar e totalizar os dados energéticos relativos às VMs.

Zhai et al. (2014) apresenta uma ferramenta chamada HaPPy, compatível com versões do kernel do Linux iguais ou superiores à 3.14. Sua funcionalidade é estimar o consumo energético de processos por meio do consumo de energia individual de suas *threads*. Considerando que uma VM é um processo, tal ferramenta pode ser utilizada para estimar também o consumo energético de máquinas virtuais. Os dados energéticos são

coletados por meio da ferramenta Linux perf, que após modificação feita pelos autores, acessa em baixo nível a interface de *software* do Intel RAPL (CORPORATION, 2015). Apesar dessa interface não prover informações energéticas sobre os cores de forma individualizada, o HaPPy utiliza uma técnica de isolamento, implementada com base nos *performance counters*, para prever o consumo energético individual das *threads*.

Li et al. (2012) propõe um modelo energético para estimar o consumo de energia de máquinas virtuais em nuvem, sem a necessidade de *hardware* especial. O modelo se aplica a *hosts* Linux que utilizem o VMware Workstation (INC., 2015). Os dados para geração do consumo energético são obtidos por meio da ferramenta Linux sar, que coleta as medidas de uso dos recursos do sistema operacional, como discos, CPU e memória. Esse solução serve tanto para estimar o consumo energético do servidor físico quanto o de suas máquinas virtuais.

Borgetto e Stolf (2014) apresentam um trabalho que visa estimar o consumo energético quando ocorrem alocações e realocações de máquina virtuais em um *data center*. Cada processador físico está alocado a uma VM e os dados energéticos são coletados por meio de um medidor de potência conectado a cada CPU real. Esse sistema roda em *hosts* Linux, utiliza o KVM e é específico para o OpenNebula (PROJECT, 2015). Não está explicito no artigo a metodologia utilizada para exibir os dados energéticos.

Colmant et al. (2015) propõe uma solução que independe de *hardware*, é específica para o KVM e trabalha integrada por meio de um sistema distribuído. Essa solução é baseada na ferramenta PowerAPI (GROUP, 2015), semelhante à solução deste trabalho. Os dados energéticos são coletados via *performance counters*, e até há uma referência sobre a possibilidade de exibição dos dados em uma interface *Web*, porém, não constam informações sobre integração com gerenciadores de nuvem.

Bertran et al. (2011) propõe uma metodologia que estima o consumo energético de máquinas virtuais e é específica para *hosts* Linux. Não há necessidade de especificidade de *hardware* e os dados energéticos são obtidos por meio da ferramenta Linux pfmon, que lê os *performance counters* referentes à CPU e memória, e os trabalha em um modelo energético.

Marcu, Tudor e Fuicu (2011) trás um método aplicado a solução de virtualização da VMware (INC., 2015). Os dados energético são obtidos por um computador extra que está ligado via Universal Serial Bus (USB) a um medidor de consumo de energia chamado Watts up (UP?, 2015). Essa dependência de *hardware* externo torna a solução apresentada inviável em larga escala.

Chengjian et al. (2013) apresenta uma solução que é específica para o hipervisor L4 (FIASCO, 2015). Os consumos de energia das VMs são estimados por um modelo energético que infere os dados a partir dos *performance counters*, que em mais baixo nível

coletam as estatísticas de uso da memória e da CPU, utilizando para isso a ferramenta Linux perf. Segundo o autor desse trabalho, o consumo energético de disco e de rede para a maioria dos servidores é praticamente pequeno e estático, não sendo tão considerável numa estimativa final.

Bourdon et al. (2011) apresenta PowerAPI (GROUP, 2015), uma ferramenta que mede o consumo energético de processos do sistema operacional hospedeiro Linux. Apesar da sua não especificidade para ambientes virtualizados, é possível utilizá-la para medir o consumo de energia de máquinas virtuais do KVM. Os dados para geração do consumo energético podem vir de meios externos como medidores de potência que interceptam o consumo de energia do servidor físico, como também a partir dos *performance counters*. Os dados coletados são aplicados no modelo energético, que irá gerar o consumo de energia do processo, ou de um grupo de processos, com base nas informações de disco, memória, rede e CPU. A ferramenta em questão está disponível para *download* e utiliza uma abordagem modular, descomplicando possíveis novas implementações. É possível utilizá-la dentro de um shell, ou como uma biblioteca de funções, dando interoperabilidade a ferramenta. Diferentemente de outras ferramentas, que precisam de intervenção para calibrar os modelos energéticos, a PowerAPI possui uma autocalibração, fazendo com que essa solução seja ideal em larga escala.

A PowerAPI, dentre as ferramentas analisadas nos trabalhos, é a que menos possui limitações e é a que mais está apta a ser implantada em cenários reais, sendo necessárias apenas algumas extensões de suas funcionalidades, para que a mesma seja utilizável em ambientes virtualizados e possa interagir com o gerenciador de nuvem OpenNebula.

4 Metodologia

Quando o assunto é *data center*, a maioria dos recursos para estimar o consumo energético está focado no *hardware*, desconsiderando os cenários reais onde a virtualização é predominante.

O sistema que este trabalho propõe está exposto na Figura 6. Os servidores *hosts* hospedam máquinas virtuais que podem ou não conter diferentes sistemas operacionais hóspedes. Cada máquina virtual é monitorada através de seu PID pela ferramenta PowerAPI, e com isso a ferramenta retorna dentro de intervalos de tempo o respectivo consumo energético em µWh. O sistema de arquivos de rede em conjunto com o sistema proposto neste trabalho se encarrega de tornar disponíveis os dados energéticos dos *hosts* para o gerenciador de nuvem, onde os dados são processados e exibidos na interface *Web* do OpenNebula.

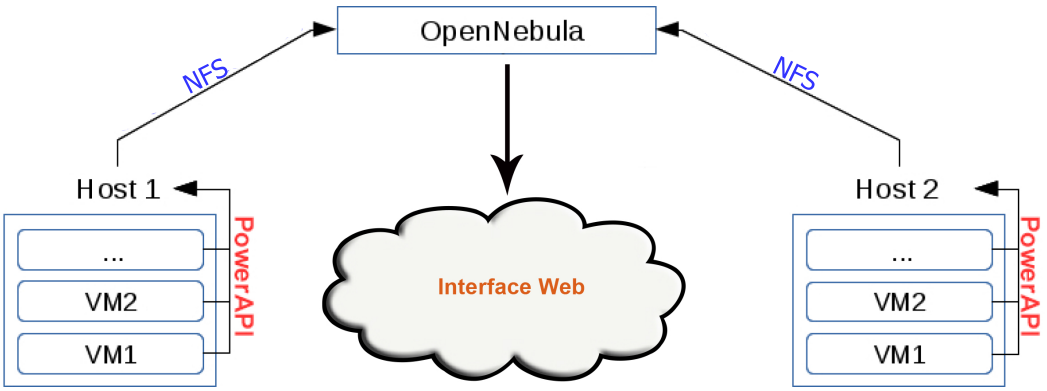

Figura 6 – Esquema de Funcionamento do Sistema de Monitoramento Energético

4.1 Configurações Iniciais dos Servidores Hospedeiros

Os servidores hospedeiros utilizados são máquinas Dell com 8GB de RAM e processador Intel i3. Antes de qualquer instalação, é necessário habilitar a tecnologia de virtualização na Basic Input and Output System (BIOS) dos servidores, visto que essa é uma prática necessária para o funcionamento do KVM. Para garantir a confiabilidade somada à rapidez de acesso aos dados, deve ser configurado, via utilitário da placa mãe, o Redundant Array of Independent Disks (RAID) na combinação 1+0.

Após as configurações iniciais do *host*, é possível passar para a instalação do sistema operacional hospedeiro.

4.2 Instalação do Sistema Operacional Hospedeiro

Por questões de estabilidade e vasta documentação, o sistema operacional escolhido para o servidor hospedeiro é o Linux. Durante a instalação, além dos pacotes que já vem por *default*, deve-se optar um ambiente sem interface gráfica. A partir do momento em que a instalação do sistema operacional hospedeiro está concluída, é possível avançar para a parte referente às aplicações.

4.3 Instalações, Configurações e Execuções das Aplicações

Na etapa das aplicações, deve ser instalada a PowerAPI para coleta dos consumos energéticos, o OpenNebula para gerir a nuvem e o libvirt para efetuar a comunicação do OpenNebula com o KVM. O sistema distribuído para comunicação entre os *hosts* e o *front end* é a própria arquitetura NFS provida pelo OpenNebula.

4.4 Viabilidade do Trabalho

Para verificar a viabilidade técnica do trabalho, foram realizados testes preliminares com as ferramentas KVM, QEMU e PowerAPI em um computador Dell Optiplex 3010, com 4GB de RAM e processador Intel i3. A instalação e execução do KVM e do QEMU foram bem sucedidas, e em razão disso, foi possível instalar uma máquina virtual com sistema hóspede Microsoft Windows XP, e monitorá-la por meio de seu PID.

5 Desenvolvimento

5.1 Tecnologia de Virtualização

É necessário que o processador do *host* possua a tecnologia Intel-VT ou AMD-V para que seja possível fazer uso da virtualização do tipo completa ofertada pelo KVM em conjunto com o QEMU.

É possível verificar a existência da tecnologia de virtualização através do comando egrep '(vmx|svm)' /proc/cpuinfo | wc -l. Se o comando retornar 0 significa que a tecnologia não está presente ou está desabilitada nas configurações da placa-mãe, incapacitando assim o *host* de ser utilizado como servidor de virtualização.

5.2 Sistema Operacional

A instalação do Sistema Operacional não é contemplada neste trabalho, pois é um processo muito particular de cada ambiente de trabalho. O único requisito para que seja possível prosseguir com a instalação e execução do Monitor Energético é que o SO seja Linux, tanto no *host* quanto no *frontend*.

5.2.1 Pacotes

Serão necessárias instalações de alguns pacotes não nativos ao sistema operacional. Essa instalação pode ser feita via gerenciador de pacotes da distribuição utilizada ou manualmente; Em caso de escolha da primeira opção, é necessário instalar o repositório do OpenNebula na lista de repositórios do Linux. No Debian é possível fazer isso através dos comandos a seguir.

```
wget http://downloads.opennebula.org/repo/Ubuntu/repo.key -O /tmp\
/repo.key
apt-key add /tmp/repo.key
echo "deb http://downloads.opennebula.org/repo/4.8/Debian/7/ sta\
ble opennebula" > /etc/apt/sources.list.d/opennebula.list
apt-get update
```

Os pacotes a serem instalados estão listados a seguir.

5.2.1.1 openssh-server

O openssh-server é o servidor responsável por receber as conexões de entrada referentes ao protocolo SSH.

5.2.1.2 qemu-kvm

O pacote qemu-kvm é necessário nos *hosts* para que o módulo do kernel referente ao KVM arbitrarize o acesso à memória e a CPU, ao passo que o QEMU emule os periféricos de entrada e saída da VM.

5.2.1.3 libvirt-bin

O pacote libvirt é um conjunto de APIs para gerenciar a virtualização no *host*. O OpenNebula utiliza por meio do virt-manager.

5.2.1.4 virt-manager

O virt-manager é um conjunto de ferramentas para ambiente gráfico ou para linha de comando que possibilita o gerenciamento do ciclo de vida das máquinas virtuais (criação, alteração, deleção, inicialização e parada).

5.2.1.5 opennebula

O pacote em questão deve ser instalado no *frontend* e é responsável por controlar o *cluster* do OpenNebula. O objetivo da clusterização nesse caso é agrupar de forma lógica *hosts*, *datastores* e redes virtuais.

5.2.1.6 opennebula-sunstone

O opennebula-sunstone é o pacote responsável por disponibilizar a interface *Web* do OpenNebula. Um socket TCP de escuta na porta 9869 será criado para que o administrador da nuvem a possa configurar e gerenciar via *browser*.

5.2.1.7 opennebula-node

Esse pacote contém as dependências necessárias para que o OpenNebula seja executado na sua versão cliente nos *hosts*.

5.2.1.8 nfs-common

O nfs-common adiciona suporte ao sistema de arquivos de rede NFS no Linux. Através desse sistema será possível compartilhar diretórios em tempo real entre *hosts* e *frontend*.

5.2.1.9 bridge-utils

Um dos requisitos para que seja possível utilizar rede nas máquinas virtuais do KVM, e por conseguinte nos recursos da nuvem do OpenNebula, é possuir uma interface

bridge no *host*. A interface *bridge* é uma abstração para que as interfaces físicas e virtuais envolvidas possam comunicar-se umas com as outras.

5.3 Instalação e Configuração de Pacotes no Host

Os pacotes a serem instalados no *host* são bridge-utils, openssh-server, qemu-kvm, libvirt-bin, virt-manager, opennebula-node e nfs-common.

A interface *bridge* deve ser configurada no sistema operacional antes de ser criada. Uma possível configuração na distribuição Debian é a seguinte.

```
auto br0
iface br0 inet static
        address 172.29.17.77
        netmask 255.255.255.0
        broadcast 172.29.17.255
        gateway 172.29.17.254
        bridge_ports eth0
        bridge_fd 9
        bridge_hello 2
        bridge_maxage 12
        bridge_stp off
```

O serviço de rede do servidor ou o próprio deve ser reiniciado para que o SO reconheça a *bridge*. O status da interface virtual e de suas interfaces físicas anexadas pode ser constatada pelo comando brctl show.

Uma das possíveis formas de montar o diretório remoto NFS do OpenNebula no *host* é adicionar uma entrada no fstab de seu Linux, conforme exemplo a seguir. É necessário que o ponto de montagem no *host* seja no diretório *home* do usuário oneadmin do OpenNebula.

```
172.29.17.78:/var/lib/one/  /var/lib/one/  nfs   soft,intr,\
rsize=8192,wsize=8192
```

Toda gerência do *host* é feitas remotamente via SSH ou Remote Procedure Call (RPC), logo, no *host* as duas configurações básicas são a da interface *bridge* e a montagem do sistema de arquivos NFS.

5.4 Instalação e Configuração de Pacotes no Frontend

Os pacotes a serem instalados no *frontend* são opennebula, opennebula-sunstone, openssh-server e nfs-common.

Da mesma maneira que foi necessário montar o sistema de arquivos NFS no diretório *home* do usuário oneadmin, será necessário fazê-lo no *frontend*. O comando a ser utilizado para isso é o a seguir.

```
172.29.17.78:/var/lib/one/  /var/lib/one/  nfs   soft,intr,\
rsize=8192,wsize=8192
```

Por padrão o servidor Web do OpenNebula possui escuta somente na interface de *loopback*. Acessos a partir de redes com faixas de IP privadas só podem ser feitos se o arquivo sunstone-server.conf for editado e a diretiva *host* for alterada para 0.0.0.0 ou para um endereço IP que esteja disponível em uma das interfaces de rede do servidor. Para que a nova configuração entre em vigor é necessário reiniciar o serviço opennebula-sunstone.

Em alguns momentos durante as configurações é necessário trabalhar com nomes ao invés de endereços IP. É aconselhável mapear os nomes dos *hosts* para seus respectivos endereços IP no arquivo /etc/hosts do *frontend*. O padrão neste trabalho é a nomenclatura opnodeN, onde N é o número do *host*.

O *frontend* do OpenNebula utiliza o protocolo SSH para execução remota de comandos nos *hosts*. Utilizar autenticação com usuário e senha para isso seria muito trabalhoso além de inseguro, pois a senha teria de ser digitada a cada acesso ou mantida em um arquivo de texto. Em razão disso a autenticação por meio de criptografia de chave pública é utilizada (TORALDO, 2012). No *frontend* deve-se logar com o usuário oneadmin e gerar uma chave RSA através do comando ssh-keygen -t rsa. Esse comando irá criar os arquivos id_rsa.pub e id_rsa no diretório oculto ssh dentro da pasta *home* do usuário oneadmin. O arquivo id_rsa.pub contém a chave pública do *frontend*, que deve ser adicionada ao fim do arquivo authorized_keys de cada *host*. Por questões de segurança, o arquivo id_rsa que contém a chave privada do *frontend* deve possuir a permissão octal 0600.

Após a configuração do SSH, deve-se adicionar os *hosts* ao *cluster* do OpenNebula. Por mais que o pacote opennebula-node esteja instalado nos *hosts*, ainda não há contato deles com o *frontend*. A adição de cada *host* ao *cluster* deve ser feita no *frontend* com o comando a seguir.

```
onehost create opnode1 -i kvm -v kvm -n dummy.
```

Por fim, deve-se informar ao OpenNebula as configurações das *bridges* para que ele a possa gerenciar. Para isso, logado ainda com o usuário oneadmin, deve-se criar o arquivo mynetwork.one no seguinte formato:

```
NAME = "private"

BRIDGE = br0

AR = [
            TYPE = IP4,
```

```
            IP = 172.29.17.63,
            SIZE = 3
]
```

Concluídas as instalações e configurações explanadas até aqui, é possível obter uma versão funcional do cluster do OpenNebula.

5.5 AddOn Monitor Energético (ME)

O AddOn Monitor Energético (ME), proposta deste trabalho e representado pela Figura 7, possibilita realizar o monitoramento energético de nuvens computacionais geridas pelo OpenNebula. Para isso, há dois módulos de funcionamento; O módulo Monitor Energético em Bash (MEB) é o responsável por coletar dados energéticos nos *hosts* e transmití-los ao *frontend*; Já o módulo Monitor Energético em Web (MEW) é uma interface Web adicionada ao OpenNebula Sunstone para que seja possível visualizar os dados coletados em formato de gráfico. A comunicação dos dois módulos se dá por meio da própria infraestrutura do OpenNebula.

5.5.1 Instalação do Monitor Energético

O arquivo ME.bz2 deve ser descompactado no diretório temporário do Linux. Da descompactação irão resultar os arquivos instalador.bash (instalador do Monitor Energético (ME)), monitor.bash (Monitor Energético em Bash (MEB)), sunstone-server.rb (Configuração do servidor Web), monitor.erb (acmew), main.js (gerador JavaScript da interface Web do OpenNebula Sunstone). A instalação do ME se dá pelo script instalador.sh, ele deve ser executado no *frontend* do OpenNebula e suas funcionalidades são fazer *download* da PowerAPI, dar permissão de execução ao script monitor.sh, copiá-lo para o diretório /var/lib/one/remotes/im/kvm-probes.d e sincronizar o *frontend* com os *hosts* por meio do comando onehost sync –force. Esse comando copia remotamente os arquivos do diretório /var/lib/one/remotes do *frontend* para o diretório /var/tmp/one dos *hosts*.

5.5.2 Monitor Energético em Bash (MEB)

Esse módulo está representado pelo script monitor.bash disponível no apêndice. O script é executado automaticamente pelo *scheduler* do OpenNebula em intervalos de tempo pré-definidos e a sua saída é escrita no arquivo /var/lib/one/dados.txt. O diretório /var/lib/one é disponibilizado via NFS em todos os servidores do *cluster* do OpenNebula.

O monitor.bash quando executado cria um arquivo de *lock* para que os valores das estimativas energéticas não sejam eventualmente alterados, já que os *hosts* do *cluster*

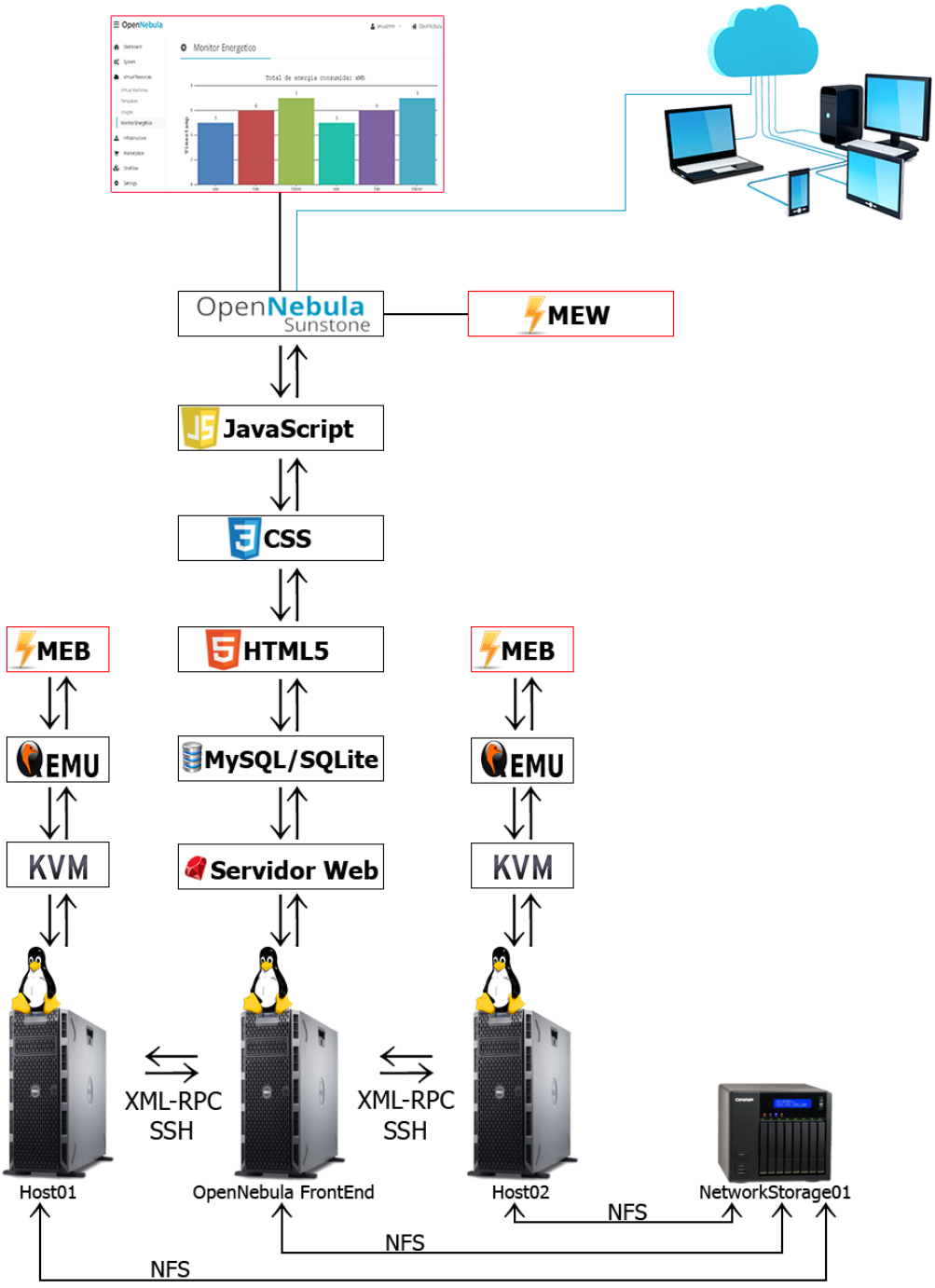

Figura 7 – Monitor Energético (ME)

executam o script e escrevem no arquivo dados.txt quase simultaneamente. Se o arquivo de *lock* não existe, o script continua sua execução e verifica quais máquinas virtuais estão em funcionamento no *host*. As informações referentes aos processos das VMs em execução são escritas no arquivo /tmp/VMsRODANDO.

Um laço de repetição lê as informações do arquivo /tmp/VMsRODANDO e para cada VM lida (cada interação), o PID e o nome da máquina virtual são armazenados

nas variável PIDVM e NOMEVM, respectivamente. As duas variáveis posteriormente são passadas por parâmetro para a função monitorEnergetico. A função é chamada junto de um & para que a cada interação ela seja executada em paralelo sem atrapalhar a continuidade do laço. Se isso não fosse feito, para cada máquina virtualde lida deveria-se esperar o tempo de execução da função para que a próxima interação do laço pudesse ser feita; Após as chamadas à função monitorEnergetico há um *wait* para sincronizá-las.

A função monitorEnergetico primeiramente executa a ferramenta PowerAPI. Os parâmetros passados para a ferramenta são definidos em constantes no começo do script, sendo elas o módulo de monitoramento procfs-cpu-simple, o arquivo temporário de escrita em /tmp/PIDVM.tmp e a duração de cerca de 18s. A execução da ferramenta irá gerar uma saída no arquivo temporário a cada segundo, sendo que cada linha contém informações referentes ao *timestamp* de execução e ao consumo energético instantâneo do processo para aquele *timestamp*.

Após coletar os dados referentes ao consumo energético do processo, há um laço de repetição dentro da função monitorEnergetico que retém o valor energético de cada linha do arquivo /tmp/PIDVM.tmp, o divide por 3600 para obter Wh e o adiciona numa variável para que ocorra o somatório dos valores encontrados em todas as linhas do arquivo. Como os valores são relativamente pequenos, para melhor exibição no gráfico da interface Web eles são convertidos para µWh. A função monitorEnergetico, antes de chegar no fim de sua execução, chama a função escreveArquivo para que o valor obtido seja escrito no arquivo /var/lib/one/dados.txt.

A função escreveArquivo utiliza o formato JSON para escrever no arquivo dados.txt (utilizado pelo OpenNebula) e o formato CSV para escrever no arquivo dados-export.csv (possibilidade de importação em outros *softwares*). O formato JSON foi adotado ao invés do XML em razão de compatibilidade com o *framework* CanvasJS (FENOPIX, 2016) utilizado para plotar os gráficos na interface *Web* do OpenNebula.

O último passo do script monitor.bash é remover o arquivo de *lock* para que uma nova instância do mesmo possa ser iniciada. Feito isso, os dados estarão prontos para serem lidos pelo Monitor Energético em Web (MEW).

5.5.3 Monitor Energético em Web (MEW)

O OpenNebula possui vários desafios quanto a modificação de sua parte *Web*. Para começar o servidor *Web* utilizado é um caso atípico onde ao invés de ser usado o Apache (FOUNDATION, 2016b) é executado um servidor *Web* via script desenvolvido com a linguagem de programação Ruby. Em razão disso e por restrição do script sunstone-server.rb, qualquer nova página HTML precisa possuir a extensão erb e deve ser previamente liberada para exibição na interface *Web*. Não há execução de um interpretador *server-side*

como o PHP (GROUP, 2016) no OpenNebula, quando há a necesidade de execução de uma *query* Structured Query Language (SQL), por exemplo, é necessário fazer uma chamada ao servidor *Web* para que o mesmo execute a SQL previamente definida em arquivos do servidor *Web*. Essa restrição junto ao fato de que há duas possibilidades de uso de sistemas de gerenciamento de banco de dados no OpenNebula, induziu a implementação deste sistema a não trabalhar com banco de dados e sim com arquivos de texto.

O *iframe* inserido na página apontada pelo menu garante a exibição de outra página denominada monitor.erb conforme mostrado na Figura 8. Essa página contém códigos HTML e JavaScript que importam os dados energéticos do arquivo /var/lib/one/dados.txt e os exibem em um gráfico de barras gerado pelo *framework* CanvasJS. O arquivo dados.txt é processado pelo método parseJSON do JQuery (FOUNDATION, 2016a) e os dados resultantes são inseridos em um vetor temporário, que por sua vez será copiado para o objeto *data* do gráfico. Feito isso, o gráfico pode ser renderizado através do método chart.render().

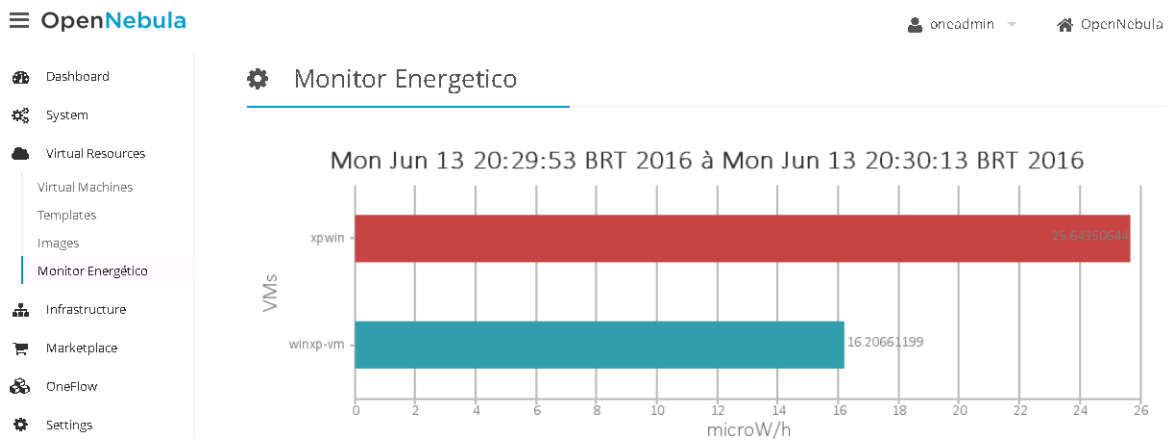

Figura 8 – Entrada referente ao Monitor Energético no menu do Sunstone

O gráfico é atualizado via JQuery em intervalos de dois segundos, podendo esse valor ser modificado se necessário.

5.6 Validação do AddOn Proposto

Para validar o sistema proposto neste trabalho, realizou-se uma estimativa com o MEB em uma máquina virtual que rodou por cerca de dez minutos o *benchmark* Sysbench (KOPYTOV, 2016). O propósito desse teste foi realizar um estresse na CPU de modo a ser possível comprovar o aumento do consumo de energia elétrica nos minutos analisados.

A Figura 9 mostra que a partir da metade do minuto vinte e nove o *benchmark* é inicializado na máquina virtual e ocorre um pico no consumo de energia elétrica. Os

valores nos minutos seguintes mantém-se em alta em relação aos valores anteriores ao início da execução do *benchmark*.

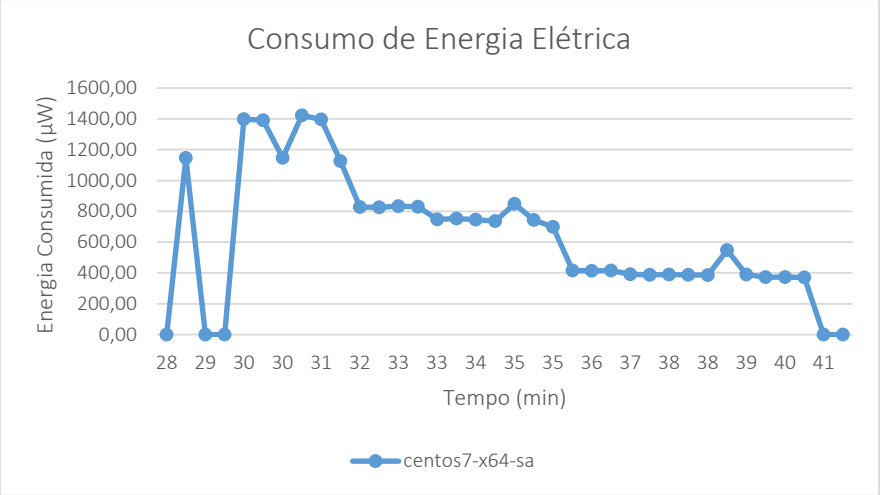

Figura 9 – Gráfico do consumo energético da máquina virtual centos7-x64-sa

6 Conclusão

O objetivo desse trabalho foi desenvolver um sistema para realizar o monitoramento energético de nuvens computacionais do OpenNebula. O sistema serve de auxílio aos administradores de nuvem para mostrar o quanto de energia elétrica está sendo consumida pelas máquinas virtuais do KVM.

Apesar da pouquíssima documentação sobre como integrar *softwares* de terceiros e sobre como construir *addons* para o OpenNebula *Sunstone*, foi possível construir o Monitor Energético proposto neste trabalho. Grande parte de sua construção foi possível por meio de engenharia reversa nos códigos já existentes e de tentativa e erro.

A computação em nuvem é uma área em constante crescimento e a tendência é que cada vez mais os serviços locais sejam migrados para a nuvem. Isso dá aos usuários e as empresas praticidade no acesso, mas com o aumento do uso de recursos ocorre também o aumento do consumo de energia elétrica. O monitoramento energético da nuvem computacional é importante justamente para trazer indicativos sobre essa situação.

Como trabalhos futuros pretende-se implementar tomadas de decisões em relação às estimativas energéticas obtidas. Pretende-se também estender as funcionalidades do Monitor Energético ao gerenciador de nuvem OpenStack.

6.1 Cronograma

As tarefas serão distribuidas através dos meses conforme a Tabela 2.

Tabela 2 – Cronograma de Atividades

Tarefas	**Ago**	**Set**	**Out**	**Nov**	**Dez**	**Jan**	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**
Revisar literatura	OK	OK	-	-	-	-	-	-	-	-	-
Estudar sobre virtualização e computação em nuvem	-	OK	OK	-	-	-	-	-	-	-	-
Pesquisar sobre eficiência energética	-	OK	OK	-	-	-	-	-	-	-	-
Realizar o esboço teórico do sistema	-	-	-	OK	-	-	-	-	-	-	-
Conclusão do trabalho para a primeira entrega	-	-	-	OK	OK	-	-	-	-	-	-
Implementar o sistema	-	-	-	-	-	OK	OK	OK	-	-	-
Realizar os testes	-	-	-	-	-	-	-	OK	OK	OK	-
Coletar os Resultados	-	-	-	-	-	-	-	OK	OK	OK	-
Finalização do trabalho para a segunda entrega	-	-	-	-	-	-	-	-	-	-	OK

Referências

VOGEL, A. et al. (Ed.). *Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack*. Grécia, 2016. Citado na página 23.

CHENGJIAN, W. et al. (Ed.). *System Power Model and Virtual Machine Power Metering for Cloud Computing Pricing*. China, 2013. Citado 2 vezes nas páginas 34 e 35.

VERSICK, D.; WABMANN, I.; TAVANGARIAN, D. (Ed.). *Power Consumption Estimation of CPU and Peripheral Components in Virtual Machines*. Germany, 2013. Citado na página 34.

ALCIOM. *PowerSpy2*. 2015. Disponível em: <https://www.alciom.com/en/products/powerspy2-en-gb-2.html>. Citado na página 25.

CORPORATION, I. *RAPL Power Meter*. 2015. Disponível em: <https://01.org/rapl-power-meter>. Citado na página 35.

BERTRAN, R. et al. (Ed.). *Energy Accounting for Shared Virtualized Environments Under DVFS Using PMC-based Power Models*. Spain, 2011. Citado 2 vezes nas páginas 34 e 35.

MARON, C. A. F. et al. (Ed.). *Avaliacão e Comparacão do Desempenho das Ferramentas OpenStack e OpenNebula*. Brasil, 2014. Citado na página 31.

ROVEDAL, D. et al. (Ed.). *Analisando a Camada de Gerenciamento das Ferramentas CloudStack e OpenStack para Nuvens Privadas*. Brasil, 2015. Citado na página 23.

COLMANT, M. et al. (Ed.). *Process-level Power Estimation in VM-based Systems*. France, 2015. Citado 2 vezes nas páginas 34 e 35.

FENOPIX. *CanvasJS: Beautiful HTML5 JavaScript Charts*. 2016. Disponível em: <http://canvasjs.com>. Citado na página 45.

FIASCO. *L4Re – L4 Runtime Environment*. 2015. Disponível em: <http://l4re.org/>. Citado na página 35.

FOUNDATION jQuery. *jQuery Foundation*. 2016. Disponível em: <https://jquery.org/>. Citado na página 46.

FOUNDATION, T. A. S. *Welcome! - The Apache HTTP Server Project*. 2016. Disponível em: <https://httpd.apache.org/>. Citado na página 45.

BOURDON, A. et al. (Ed.). *Linux: Understanding Process-Level Power Consumption*. France, 2011. Citado 3 vezes nas páginas 23, 34 e 36.

GROUP, S. R. *PowerAPI*. 2015. Disponível em: <http://www.powerapi.org/>. Citado 2 vezes nas páginas 35 e 36.

GROUP, T. P. *PHP: Hypertext Preprocessor*. 2016. Disponível em: <http://php.net/>. Citado na página 46.

BOHRA, A. E. H.; CHAUDHARY, V. (Ed.). *VMeter - Power Modelling for Virtualized Clouds*. USA, 2010. Citado 2 vezes nas páginas 33 e 34.

LI, Y. et al. (Ed.). *An Online Power Metering Model for Cloud Environment*. China, 2012. Citado 2 vezes nas páginas 34 e 35.

BORGETTO, D.; STOLF, P. (Ed.). *An Energy Efficient Approach to Virtual Machines Management in Cloud Computing*. France, 2014. Citado 2 vezes nas páginas 34 e 35.

INC., V. *VMware Virtualization for Desktop and Server*. 2015. Disponível em: <http://www.vmware.com/>. Citado na página 35.

JAMSA, K. *Cloud Computing*. [S.l.], 2012. Citado na página 29.

JANG, M. *Security Strategies in Linux Platforms and Applications*. [S.l.], 2010. Citado na página 27.

KOPYTOV, A. *sysbench in Launchpad*. 2016. Disponível em: <https://launchpad.net/sysbench>. Citado na página 46.

NETO, M. V. de S. *Computação em Nuvem: Nova Arquitetura de TI*. [S.l.], 2015. Citado na página 28.

PEPPLE, K. *Deploying OpenStack*. [S.l.], 2011. Citado na página 29.

PETERSEN, R. *Fedora 14: Administration and Security*. [S.l.], 2010. Citado na página 27.

PROJECT, O. *OpenNebula | Flexible Enterprise Cloud Made Simple*. 2015. Disponível em: <http://opennebula.org/>. Citado 3 vezes nas páginas 30, 33 e 35.

SCHULZ, G. *The Green and Virtual Data Center*. [S.l.], 2009. Citado na página 23.

KANSAL, A. et al. (Ed.). *Virtual Machine Power Metering and Provisioning*. USA, 2010. Citado na página 34.

VOGEL, A. et al. (Ed.). *HiPerfCloud: Um Projeto de Alto Desempenho em Nuvem*. Brasil, 2015. Citado na página 29.

ROVEDAL, D.; VOGEL, A.; GRIEBLER, D. (Ed.). *Understanding, Discussing and Analyzing the OpenNebula and OpenStack's IaaS Management Layers*. Brasil, 2015. Citado na página 29.

TANENBAUM, A. S. *Sistemas Operacionais Modernos*. [S.l.], 2010. Citado 3 vezes nas páginas 25, 26 e 27.

MARCU, M.; TUDOR, D.; FUICU, S. (Ed.). *Power Consumption and Temperature Measurement of Virtualization Solutions*. France, 2011. Citado 2 vezes nas páginas 34 e 35.

TORALDO, G. *OpenNebula 3 Cloud Computing*. [S.l.], 2012. Citado 3 vezes nas páginas 28, 30 e 42.

UP?, W. *Watts up? Pro*. 2015. Disponível em: <http://www.wattsupmeters.com>. Citado na página 35.

ZHAI, Y. et al. (Ed.). *HaPPy - Hyperthreaded-aware Power Profiling Dynamically*. USA, 2014. Citado na página 34.

Apêndices

APÊNDICE A – monitor.bash

```
#!/bin/bash
#Monitor Energético em Bash (MEB) - Trabalho de Conclusão de Curso - Ciência da Computação - UNIPAMPA - Campus Alegrete
#2016-04-02 - Criação
#2016-06-13 - Atualização
#Feito por Raul Leiria (rdleiria@gmail.com)


#Constantes
DESTINOARQFINALWHVM="/tmp"
PAPIPATH="/var/lib/one/powerapi/bin";
PAPIMODULO="procfs-cpu-simple"
PAPIFREQUENCIA="1000"
PAPIDURACAO="18" #Em segundos
NOMENODE=`hostname`
ARQUIVOJSON="/var/lib/one/dados.txt"
ARQUIVOJSONTMP="/tmp/dados.txt"
ARQUIVOEXPORT="/var/lib/one/dados-export.csv"
ARQUIVOLOGSCRIPT="/tmp/monitor.log"
ARQUIVOSAIDACHAMREMOTA="/tmp/sshout"
IPFE="172.29.17.78"
USERFE="oneadmin"
PORTASSH="22"
TIMESTAMP="0"


#Função para converter o formato timeStamp da PowerAPI para um timestamp legível
PAPI2TSLEGIVEL(){
        DATAUNIXPAPI="$1"
        #DATAUNIXTS=`echo "$DATAUNIXPAPI" | head -c 10` #No formato cru da PAPI o timestamp vem com 13 dígitos
        DATALEGIVEL=`date -d "@$DATAUNIXTS"`
        echo "$DATALEGIVEL"
}


#Função chamada remota ao FE via SSH
chamadaRemota(){
        CMD="$1"
        ssh -qp $PORTASSH $USERFE@$IPFE "$CMD" > "$ARQUIVOSAIDACHAMREMOTA"
}


#Função para obter o nome real da máquina virtual
obterNomeVM(){
        LINHAVM="$1"
        VNUMBER=`echo "$LINHAVM" | head -n 1 | sed "s/^\ //g" | cut -d " " -f18 | cut -d "-" -f2`

        cat "$ARQUIVOSAIDACHAMREMOTA" | grep " $VNUMBER oneadmin" | cut -d " " -f9
}


#Função para registrar o timestamp do log
LOG(){
        echo `date +%Y-%m-%d--%H-%M-%S` "-" `hostname` "---> $1" >> $ARQUIVOLOGSCRIPT
}


#Função responsável por inserir os dados energéticos no arquivo ARQJSON via SSH
escreveArquivo(){
        NOMEVM="$1"
        VALOR="$2"
        TSINICIO="$3"
        TSFIM="$4"
        DEPOISDE="{"

        #Se a variável está vazia, significa que o monitoramento naquele período é 0
```

```
        if [ -z "$VALOR" ]; then
                VALOR=0;
        fi;

        #Se o arquivo temporário JSON não existe, então ele é criado
        if ! [ -f "$ARQUIVOJSONTMP" ]; then
                echo -e "[\n]" > "$ARQUIVOJSONTMP"
        fi;

        #Verifica se a VM já consta no arquivo
        TMPTS=`date`;
        cat $ARQUIVOJSONTMP | grep -q "\"$NOMEVM\""
        if [ $? -eq 0 ]; then
                #Sim
                sed "s/{\"nome\": \"$NOMEVM\",\"float\": .*,\"timestamps\": \".*/{\"nome\": \"$NOMEVM\",
\"float\": $VALOR,\"timestamps\": \"$TIMESTAMP à $TMPTS\"},/g" -i $ARQUIVOJSONTMP
        else
                #Não
                sed "s/\[/\[\n\{\"nome\": \"$NOMEVM\",\"float\": $VALOR,\"timestamps\": \"$TIMESTAMP à
$TMPTS\"\},/g" -i $ARQUIVOJSONTMP
        fi;

        #Remove a vírgula da última linha
        SAIDA=`tail -n 2 $ARQUIVOJSONTMP | head -n 1 | sed "s/\},/\}/g"`
        sed "s/$SAIDA,/$SAIDA/g" -i $ARQUIVOJSONTMP

        #Atualiza arquivo JSON através do usuário oneadmin. Outro usuário, inclusive, não tem permisão de
escrita
        cat $ARQUIVOJSONTMP > $ARQUIVOJSON

        #Exporta os dados para guardar o historico e ser possivel plotá-los em ferramentas externas
        echo "$NOMEVM;$VALOR;$TSINICIO;$TSFIM" >> $ARQUIVOEXPORT
}


#Função responsável por monitorar o consumo energético da VM
monitorEnergetico(){
        #PID da VM vem como parâmetro dessa função
        PIDVM="$1"

        #Nome da VM
        NOMEVM="$2"

        #É necessário estar no diretório do binário do PowerAPI por questões de comptabilidade
        cd $PAPIPATH

        #Limpar arquivo
        cat /dev/null > /tmp/$PIDVM.tmp

        #Monitoramento do PID da VM via PAPI
        ./powerapi modules $PAPIMODULO monitor --frequency $PAPIFREQUENCIA --console --pids $PIDVM --
file /tmp/$PIDVM.tmp duration $PAPIDURACAO > /dev/null

        #Início somatório
        #
        #
        SOMATORIOWH="";
        TSINICIO=0;
        TSFIM=0;
        C=0;

        #Laço de repetição que a cada interação processa uma linha do arquivo com os dados energeticos
referentes ao PID da VM
        while read -r l; do
                #Pega o valor em joule/s da linha lida
                JOULEPSEGUNDO=`echo "$l" | cut -d "=" -f6`;
```

```
                #Se o valor (j/s) é 0 então ele não entra no somatório. O if a seguir faz a lógica
inversa disso.
                if ! [ $JOULEPSEGUNDO == 0.0 ]; then
                        #Acúmulo do somatório dos valores em j/s (j/s = watt), sendo cada parcela da soma
dividida por 3600 para obter-se watt/h
                        SOMATORIOWH+="$JOULEPSEGUNDO/3600 + ";
                fi;
        done < "/tmp/$PIDVM.tmp";

        #Timestamps de início e de fim das estimativas no tempo estipulado na constante PAPIDURACAO
        #Por mais que o primeiro e o último valor possam ser zerados, o período de coleta ainda os
compreende e deve iniciar e terminar respectivamente por eles
        TSINICIO=`head -n 1 /tmp/$PIDVM.tmp | cut -d "=" -f3 | sed "s/;targets//g" | head -c 10`
        TSFIM=`tail -n 1 /tmp/$PIDVM.tmp | cut -d "=" -f3 | sed "s/;targets//g" | head -c 10`

        #Remover o último sinal de soma
        TOTALEMWH=`echo $SOMATORIOWH | rev | sed "s/+ //" | rev`

        #Calculo do somatório e do ajuste da unidade para micro watt/h
        TOTALEMWH=`echo $SOMATORIOWH | rev | sed "s/+ //" | rev | bc -l`
        TOTALEMMICROWH="$TOTALEMWH/10^(-6)"
        TOTALEMMICROWH=`echo $TOTALEMMICROWH | bc -l`;

        #Volta ao diretório inicial deste script
        cd -

        #Escrita no arquivo temporario JSON
        escreveArquivo "$NOMEVM" "$TOTALEMMICROWH" "$TSINICIO" "$TSFIM"
        #
        #
        #Fim somatório
}


#Será criado um arquivo de lock para cada instancia desse script, sendo uma instancia para cada node
#O lock é necessário para que não haja a situação onde o OpenNebula tente inicializar esse script
novamente antes dele ter terminado todos os monitoramentos da chamada anterior
#Separador no log
echo -e "\n" >> $ARQUIVOLOGSCRIPT
if ! [ -f .$NOMENODE.lock ]; then
        #Cria arquivo de lock
        touch .$NOMENODE.lock
        LOG "Arquivo de lock criado."

        #Sinaliza que o script começou o trabalho
        #echo "------------------------ MONITOR ENERGÉTICO EM BASH (MEB) ------------------------";
        #date
        #echo " ---> O script está sendo executado, aguarde cerca de $PAPIDURACAO segundos por favor...";

        #Listar VMs rodando no momento e colocar isso no arquivo /tmp/VMsRODANDO
        ps ax | grep qemu-system-x86_64 | grep Sl > /tmp/VMsRODANDO
        LOG "Listagem de VMs rodando."

        #Só prossegue com a execução se há VMs rodando no host
        if [ -s /tmp/VMsRODANDO ]; then
                LOG "Há VMs rodando."
                #Nome das VMs
                chamadaRemota "onevm list | tail -n 2"

                #Timestamp
                TIMESTAMP=`date`;

                #Para cada VM sera executado um monitor energetico
                while read -r v; do
                        #PID da VM
                        PIDVM=`echo "$v" | cut -d " " -f1`;
```

```
                        #Nome da VM
                        NOMEVM=`obterNomeVM "$v"`

                        #Monitor Energetico para o PID encontrado nessa interacao
                        monitorEnergetico "$PIDVM" "$NOMEVM" &

                        #Sincronização dos processos
                        wait
                done < "/tmp/VMsRODANDO";
        else
                LOG "Não há VMs rodando."
        fi;

        #Tira o lock
        rm .$NOMENODE.lock
        LOG "Remoção do arquivo de lock."

        #Data término
        #date
        #echo "------------------------ MONITOR ENERGÉTICO EM BASH (MEB) ------------------------";
else
        LOG "O arquivo de lock é existente, há algum host executando o monitor. Encerrando execução do
script...."
fi;
```

APÊNDICE B – monitor.erb

```
<!DOCTYPE HTML>
<html>

<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.2/jquery.min.js"></script>
<script src="http://canvasjs.com/assets/script/canvasjs.min.js"></script>
<script type="text/javascript">
        jQuery(document).ready(function($){
                setInterval(function(){
                        $.get('dados.txt', function(data){
                                var dados = jQuery.parseJSON(data);
                                var dps = [];
                                        for(a in dados){
                                                dps.push(
                                                        {
                                                                label: dados[a].nome,
                                                                y: parseFloat(dados[a].float),
                                                        }
                                                );
                                        }

                                var chart = new CanvasJS.Chart("chartContainer",{
                                        theme: "theme1",
                                        title:{
                                                text: dados[a].timestamps
                                        },
                                        axisY: {
                                                title: "microW/h"
                                        },
                                        axisX: {
                                                title: "VMs"
                                        },
                                        legend:{
                                                verticalAlign: "top",
                                                horizontalAlign: "centre",
                                                fontSize: 18,
                                        },
                                        data:[{
                                                type: "bar",
                                                showInLegend: false,
                                                legendMarkerType: "none",
                                                indexLabel: "{y}",
                                                dataPoints: dps
                                        }]
                                });

                                chart.render();
                        });
                },2000)

        });
</script>
</head>

<body>
<div id="chartContainer" style="height:300px; width:100%;"></div>
</body>

</html>
```