Uma ferramenta para análise do impacto da migração de máquinas virtuais

Daniel S. Camargo, Guilherme P. Koslovski

[1]Programa de Pós-Graduação em Computação Aplicada (LabP2D/PPGCA – DCC)
Universidade do Estado de Santa Catarina (UDESC) – Joinville/SC – Brasil
daniel@colmeia.udesc.br,
guilherme.koslovski@udesc.br

Resumo. *A migração de máquinas virtuais (MVs) é um mecanismo utilizado para o gerenciamento de sistemas virtualizados, como a computação em nuvem, que possibilitar mover uma MV instanciada em uma determinada máquina física (MF) para outra. Este gerenciamento normalmente é utilizado para balanceamento de carga ou para consolidação. Enquanto o balanceamento de carga visa equilibrar o uso dos equipamentos físicos, a consolidação de MVs possibilita concentrar a maior quantidade de MVs no menor número servidores, visando otimizar o uso de recursos. A estratégia de consolidação exige um intenso processo de migração de MVs entre os servidores, resultando em aumento no tráfego de dados na rede e levando ao aumento do consumo dos demais equipamentos. Neste sentido, o presente trabalho visa aplicar uma solução de monitoramento desenvolvida para mensurar o impacto da migração nos equipamentos de rede, com a execução de uma suíte de testes automatizada e agnóstica à plataforma de nuvem. Os resultados mostram que há uma redução de 21,4% na largura de banda disponível, causando até 43 retransmissões de pacotes por segundo e um incremento de 14 vezes o tempo de ida e volta de um pacote ICMP. No entanto, é utilizada nos testes apenas a migração live, e não foi observado nenhuma indisponibilidade para o usuário. Estes resultados indicam que pode haver um efeito rebote na estratégia de consolidação de MVs que baseiam-se em migração.*

1. Introdução

O paradigma de computação em nuvem evidencia-se por fornecer recursos computacionais sob demanda, de modo elástico, escalável e com garantias de disponibilidade [Mell and Grance 2011]. Estas garantias são estabelecidas por acordos entre o inquilino (proprietário do serviço) e o provedor de nuvem, em termo conhecido como *Service Level Agreement* (SLA) [Mosa and Paton 2016]. A disponibilidade é a probabilidade de um sistema ser capaz de fornecer serviços quando requisitado, em qualquer ponto no tempo e durante um período de tempo determinado, sendo especificado pela razão entre o tempo disponível e o tempo total de execução [Rohani and Roosta 2014]. Tipicamente as SLAs de provedores de nuvens públicas especificam uma disponibilidade de três até noves (entre 99,9% a 99,95%), equivalente entre 21 a 43 minutos ao mês de interrupção nos serviços. Garantir os requisitos básicos em um ambiente de computação em nuvem, requer manter inúmeras MFs ativas, que muitas vezes permanecem subutilizadas enquanto aguardam novas instâncias [Bala and Devanand 2016], e isto resulta em um elevado consumo de energia elétrica.

Para reduzir o impacto do consumo de energia em infraestruturas de computação em nuvem, *i.e.*, Data Centers (DCs), a consolidação de MVs destaca-se por ser uma das principais estratégias encontradas na literatura [Beloglazov and Buyya 2012, Ahmad 2015, Hameed et al. 2016]. Esta estratégia visa alocar as MVs no menor número possível de MFs, possibilitando que as MFs subutilizadas sejam desativadas. Esta estratégia também possibilita tornar o uso mais eficiente dos recursos físicos disponíveis no DC, como memória, processamento, armazenamento e rede. Todavia, aplicar uma abordagem de consolidação rígida, requer um grande número de migrações para a distribuição de MVs entre as MFs e pode implicar em uma série de violações do SLA no requisito de disponibilidade [Beloglazov 2013]. Em uma estratégia de consolidação de MVs normalmente há um *trade-off* entre a redução do consumo de energia[1] e as violações do SLA, conhecido como *trade-off* energia-SLA.

A estratégia de consolidação utiliza dois principais mecanismos: o posicionamento inicial otimizado e a migração de MVs [Abdelsamea et al. 2017]. Enquanto o posicionamento inicial visa selecionar a MF mais adequada para instanciar uma nova MV, a migração consiste em mudar uma MV pré-alocada para outra outra MF;. O mecanismo de migração também é utilizado para outros procedimentos, como a manutenção programada, balanceamento de carga e redimensionamento de recursos virtuais [Deshpande et al. 2012]. Na prática, a migração de uma MV baseia-se na transferência de um conjunto de processos, com suas páginas de memória e dados persistidos em volumes, entre as MF de origem e de destino.

Para o presente trabalho, define-se dois tipos básicos de migração: a *Live Migration* e a *Cold Migration* [Medina and García 2014]. A *Live Migration* consiste em transferir uma MV enquanto esta mantém-se em execução, necessitando, portanto, de uma carga maior de sincronização entre as MFs. Isto possibilita minimizar o tempo de indisponibilidade, mas demanda maior processamento para a sincronização, utiliza excessivamente a rede e pode causar inconsistências nas MVs (falhas). Quando a arquitetura da nuvem conta apenas com um sistema de armazenamento local, pode-se utilizar a modalidade de *Live Migration* por blocos, indicado para MVs com tamanhos reduzidos (*flavors* e volumes menores). Por sua vez, na *Cold Migration* a MV é suspensa, o que causa um maior tempo de indisponibilidade da MV e implica em violações do SLA [Ahmad 2015], mas possibilita utilizar menos recursos e causa menos falhas de consistência. Segundo a literatura, quando há um número excessivo de migrações, a disponibilidade é o primeiro requisito a ser afetado, devido a maior frequência de troca de contexto das MVs.

Neste contexto, a presente pesquisa consiste em investigar os efeitos que uma estratégia de consolidação pode causar nas taxas de disponibilidade, avaliando o tempo de interrupção com uma granularidade individual para cada inquilino. O monitoramento utilizado para a realização do conjunto de testes, permite desenvolver uma abordagem de consolidação que considera a disponibilidade de cada inquilino como um limiar para realizar as migrações. Isso permite obter a redução do consumo de energia com a consolidação enquanto mantém a disponibilidade individual dos inquilinos, sendo realizadas as migrações de MVs até que os limiares de disponibilidade sejam atingidos. Para o presente trabalho, é inicialmente realizado uma análise do consumo de recursos ao re-

[1]Denota-se que o termo eficiência energética é utilizado apenas quando há a efetiva redução do consumo de energia, mas mantendo-se o mesmo desempenho anteriormente observado.

alizar migrações de MVs em um ambiente de nuvem real baseado em OpenStack. Deste modo, este trabalho está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados, levantando as principais discussões sobre o impacto da consolidação e do consumo da migração. A proposta de pesquisa dos autores e o o modelo de estimativa de consumo são apresentados na Seção 3, que é efetivamente aplicada e testada na Seção 4. Por fim, os dados coletados são discutidos na Seção 5.

2. Trabalhos correlatos

Existem diferentes estratégias para manter a eficiência em ambientes de computação em nuvem, destacando-se o escalonamento de tarefas, balanceamento de carga, alocação e compartilhamento de recursos e a consolidação de MVs. Segundo a literatura [Medina and García 2014], [Ahmad 2015], [Beloglazov et al. 2012], [Patel and Vaghela 2016], [Hameed et al. 2016] e [Xu et al. 2016], a estratégia da consolidação de MVs possui resultados significativos, pois podem alcançar até 40% de redução no consumo de energia servidores. De modo geral, deve-se observar a eficiência energética fornecida pela solução em sua totalidade, evitando o efeito rebote na eficiência. Na consolidação, o efeito rebote ocorre quando há uma descompensação no desempenho em outras partes do sistemas, *e.g.*, sobrecarga nos recursos básicos (CPU, rede, memória), aumento da latência de serviço, além das próprias as violações no SLA. Neste sentido, os trabalhos relacionados limitam-se aos que evidenciam o impacto da migração de MVs quanto aplicada na estratégia de consolidação de MVs.

Um dos primeiros trabalhos a abordar o gerenciamento de recursos no contexto de virtualização em data centers, é de Nathuji and Schwan 2007. Os autores propõem uma arquitetura de gerenciamento de recursos organizados entre políticas locais e políticas globais. A nível local, a solução prioriza as estratégias de gerenciamento de MVs, e a nível global a solução obtém as informações sobre a alocação de recursos das MFs e aplica as políticas para decidir se o posicionamento da MV precisa ser adaptado. Todavia, a solução não automatiza o gerenciamento de recursos a nível global e, devido à época os autores não estavam considerando nenhuma plataforma de gerenciamento de nuvem.

Beloglazov 2013 considera as técnicas de migração e posicionamento para realizar a consolidação, realizado tanto por simulação quanto ao OpenStack,. O autor aplica o problema do empacotamento (Bin Packing) para desenvolver uma solução heurística, e considera as violações do SLA como métrica apenas observando a sobrecarga e a degradação do desempenho nas migrações. Todavia, são desconsiderados os requisitos de disponibilidade, devido ao seu objetivo de encontrar a solução ótima para a consolidação.

Gelbukh 2014 em seu trabalho, avalia a escalabilidade de uma versão do OpenStack desenvolvida pela Mirantis, que é uma das maiores contribuidoras para o código fonte do OpenStack [Gelbukh, 2014]. Mais especificamente, o trabalho avalia a quantidade de requisições simultâneas para a criação de MVs que uma infraestrutura OpenStack consegue atender. Os resultados mostram que a infraestrutura utilizada nesse trabalho consegue servir 250 requisições paralelas de criação de MVs. Além disso, foi possível instanciar 75.000 MVs na infraestrutura estudada. Apesar de os resultados serem dependentes do hardware dos servidores e da rede da infraestrutura, os resultados mostram que é possível alcançar um alto nível de escalabilidade no OpenStack.

Apesar de os trabalhos citados anteriormente consistirem em experimentos em

massa, estressando diversos componentes do OpenStack, esses não realizam uma análise da sobrecarga de rede gerada pelas migrações de MVs. Enquanto diverso trabalhos tratam a consolidação de MVs como uma ferramenta para auxiliar no uso eficiente de recursos, como energia, memória e CPU, este artigo foca na consolidação como um mecanismo que realiza um alto número de migrações de MVs, podendo causar o efeito rebote.

3. Uma proposta de avaliação do impacto das migrações de MVs

A solução proposta tem por objetivo avaliar o impacto das migrações de MVs como principal mecanismo da estratégia de consolidação de MVs, possibilitando quantificar seu efeito nas taxas de disponibilidade e na utilização de recursos de rede. Com base nos trabalhos correlatos, o presente trabalho parte do pressuposto que é necessário um grande número de migrações para obter um alto grau de consolidação, e que este alto número de migrações pode ocasionar em indisponibilidade para os usuários. Neste sentido, foi desenvolvida uma solução que permite quantificar o tempo em que uma MV fica indisponível para o usuário, durante o tempo em que é realizada a migração. Para ser uma solução agnóstica às plataformas de computação em nuvem, seu desenvolvimento é feito utilizando o conceito de camadas, que permite obter alto acoplamento e baixa coesão. Para isso, em uma das camadas é utilizado apenas as *Application Programming Interface*s (APIs) fornecidas por estas plataformas, permitem a integração com linguagem Shell script e o interpretador Bash. Deste modo, possibilita-se aplicar a solução em diferentes plataformas de nuvem privadas, como o OpenStack[2], CloudStack[3], Eucalyptus[4] e OpenNebula[5].

Como aplicações de base, são utilizados o *iperf3*[6] e o *ping* nativo do GNU/Linux. O *iperf3* que permite desempenhar testes e medições de vazão de rede com pacotes TCP e UDP, e fornece como métricas: o tamanho da janela do cliente para o controle de congestionamento, quantidade de pacotes retransmitidos, taxa de transferência e largura de banda. O uso destas métricas é útil para verificar a influência que a migração causa na largura de banda da rede, e também o número de pacotes retransmitidos. Por sua vez, o *ping* utiliza o protocolo ICMP para desencadear um *ECHO_RESPONSE* de um *host* ou *gateway* e permitir mensurar o *Round-Trip Time* (RTT) (tempo de ida e volta) de cada pacote.

A Figura 1 mostra a arquitetura da solução, que pode ser executada em um computador externo ao domínio da nuvem (denominado de Manager (1)) com acesso ao IP flutuante da instância alvo da migração (MV-a) e direitos administrativos ao controlador da nuvem (3). Como parâmetros da execução deve-se fornecer a quantidade de testes que serão executados para cada tipo de migração e o IP da MV-a a ser migrada. Apenas com o IP da MV-a é possível obter seu identificador primário (ID) da instância, indispensável para viabilizar sua migração. Salienta-se que MV-a deve ter o *iperf3* executando no modo servidor, sem necessidade de gerar *logs* pois os mesmos dados são fornecidos pelo *iperf3* cliente. Nesse momento, no Manager são executadas as aplicações de base *ping* e *iperf3* (no modo cliente) para gerarem os *logs* das ações, utilizando o *timestamp* do momento de

[2]Disponível em: `http://www.openstack.org`.
[3]Disponível em `http://cloudstack.apache.org`.
[4]Disponível em `http://open.eucalyptus.com`.
[5]Disponível em `http://opennebula.org`.
[6]Disponível em: `https://iperf.fr`.

cada execução como o nome do teste atualmente executado. Antes de migrar, são mapeados todos nós Computes (4) disponíveis na nuvem e aptos a receberem a MV-a. A solução permite utilizar todos os tipos de migração fornecidos pela solução de nuvem, desde que devidamente mapeadas e adicionadas ao código, na camada de tradução (2). Para cada migração, é gerado um *log* do tempo inicial e final da migração, bem como uma lista com *timesteps* que mostram o percentual em que está migração. No final da migração, verifica-se o estado da MV-a no host de destino e aplica-se uma correção se necessário. Por fim, os *logs* são analisados por um conjunto de ferramentas de expressão regular e com o interpretador matemático R[7]. para que sejam gerados automaticamente os dados sobre todos os testes de todas as migrações. A saída final são os gráficos que apontam o comportamento das migrações realizadas. Todos estes passos estão descritos na Figura 1

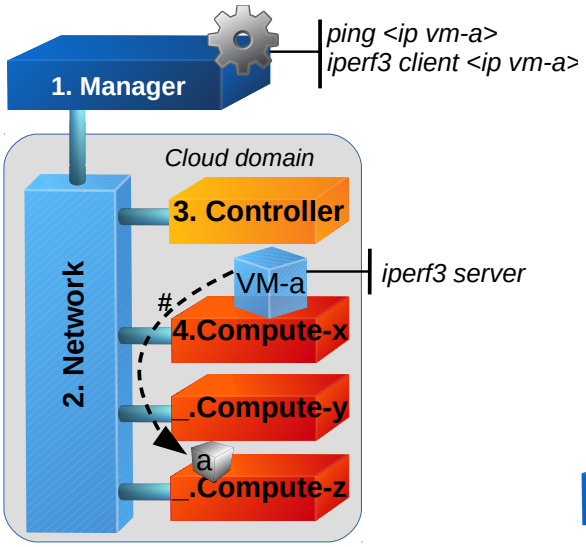

Figura 1. Sequência da solução.

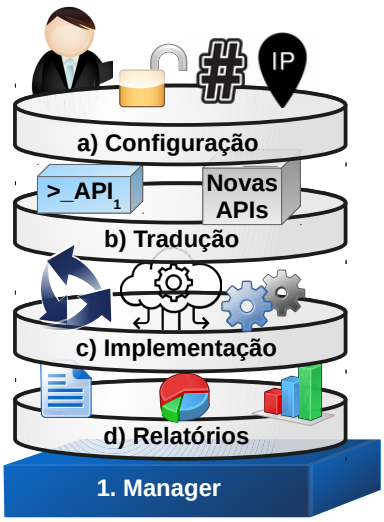

Figura 2. Desenvolvimento em camadas.

Baseando-se no conceito de desenvolvimento em camadas, a solução proposta permite maior adaptabilidade às possíveis modificações que as APIs e as aplicações de base possam vir a sofrer. São consideradas quatro camadas básicas, sendo a primeira o tratamento dos parâmetros de entrada, como a autenticação, quantidade de testes e algum identificador da instância alvo (IP, nome, ID, etc.). Em uma segunda camada há uma biblioteca de chamadas API existentes para o algoritmo de testes, implementados na terceira camada. É nessa troca entre as camadas dois e três que a solução torna-se agnóstica às plataformas de nuvem. Por fim, a saída dos dados coletados através de relatórios que informam os valores obtidos através de médias, medianas, interquatis, desvio padrão, mínimo e máximo. Toda esta solução foi aplicada em um estudo de caso em ambiente real, permitindo verificar suas funcionalidades.

4. Estudo de caso

Todos os testes são aplicados no Laboratório de Processamento Paralelo e Distribuído (LabP2D) da Universidade do Estado de Santa Catarina (UDESC), que consiste em um DC de pequeno porte que oferece recursos humanos e físicos para realização de atividades de pesquisa, ensino e extensão à comunidade acadêmica da UDESC.

[7]Disponível em `https://cran.r-project.org/`.

O ambiente de testes é constituído por cinco computadores desktops HP AMD dual core, com 8 GB de memória RAM, 500 GB de armazenamento (apenas local) e duas interfaces de rede Gigabit cada, conectadas por um switch gerenciável D-Link DGS-3100. Todos estes são considerados equipamentos comuns (*commodities*) de fácil acesso, permitindo maior reprodutibilidade dos experimentos. Todos os testes são realizados na plataforma de nuvem OpenStack, uma solução de nuvem computacional de código aberto modular e escalável, na versão Newton estável, configurada e gerenciada através do Fuel . O Fuel é uma ferramenta de código aberto que permite automatizar a implantação do OpenStack, executar testes e adicionar diversos tipos de extensões. Em relação à Figura 1, o Fuel é o Manager (1). Todos os nós de computação utilizam o *hypervisor* KVM com libvirt. Todas as MVs usadas para migração são homogêneas, com a distribuição Ubuntu 16.04 em extensão *qcow2*, e configuradas com 1 GB de memória RAM e 1 VCPU. A nuvem de testes não possui um sistema armazenamento distribuído, *i.e.*, seu armazenamento é apenas local, o que impossibilita utilizar todos os tipos de migração oferecidos pelo OpenStack.

O plano de testes foi desenvolvido de modo a permitir a reprodutibilidade dos experimentos. As métricas utilizadas consideram o tempo de execução das migrações e o RTT do *ping* em segundos. Para o *iperf3*, é considerado o número de pacotes retransmitidos para avaliar a perda de pacotes e a largura de banda. Para a execução deste cenário, são executadas vinte migrações do tipo *live*, sendo analisados os valores de mínimo, máximo, média, mediana, interquartis e desvio padrão.

Com a realização dos testes, deseja-se quantificar a interferência que um conjunto de migrações de MVs pode causar na infraestrutura de rede. Esta verificação indica se pode haver algum problema de indisponibilidade para o usuário da MV, enquanto a migração ocorre. Deseja-se também obter o tempo em que uma migração MV deve executar entre as MFs de origem e destino até terminar a sincronização.

5. Análise dos resultados

A sumarização de todos os dados é realizada automaticamente com o auxílio da aplicação de estatística *R*, que permite obter os valores de mínimo, máximo, média, mediana e desvio padrão de cada métrica. Adicionalmente é realizada uma análise sobre a quantidade de falhas de cada métrica, possível apenas pela aplicação de expressão regular em palavras-chave nos *logs*. A Tabela 1 mostra os resultados da execução de vinte testes, utilizando apenas a migração *live*.

Métricas	Migração (seg)	Ping (seg)	Retransmissão(un)	Banda (Mb/s)
Média	167	10,61	1,06	93,04
Mediana	138	6,71	0	93,9
Desvio padrão	32	18,25	5,14	4,79
Mínimo	129	2,32	0	73,1
Máximo	576	165	43	102
Qtd Falhas	0	0	129	21,4%

Tabela 1. Resultados dos testes.

[7]Fuel Mirantis, disponível em: `https://www.fuel-infra.org`.

A Tabela 1 evidencia que, embora haja picos de tempo de migração (valor máximo alto), o desvio padrão aponta que há um valor estável para os tempos de migração. Durante os testes de migração, não houve nenhuma falha observada nos *logs*. Deste modo, infere-se que a migração do tipo *live* não causa indisponibilidade aos usuários de um serviço hospedado em uma máquina que está em migração. Em relação aos tempos disponibilizados pelo *ping*, observa-se uma maior variação, entre 2 a 165 segundos, com esse valor máximo observado nos estágios finais da sincronização da migração (entre 98 a 100%). A quantidade de pacotes retransmitidos por cada segundo apresentaram um máximo de 43 retransmissões, que acumularam um total de 129 falhas durante os testes. Observando a média e o desvio padrão destas retransmissões, se conclui que são falhas esparsas, e em sua grande maioria em zero, Por fim, a largura de banda é afetada em até 21,4% quando relacionado o valor médio com o menor valor obtido nos *logs*. Isto ocorre devido à utilização da rede para a transferência dos dados da MV que está sendo migrada.

Com base nos resultados, observa-se que a migração interfere em algumas métricas simples da rede física de uma nuvem. Deste modo, conclui-se que uma estratégia de consolidação que se baseia em altas taxas de consolidação pode afetar demasiadamente a infraestrutura física do DC de nuvem.

6. Considerações finais

Atualmente, diversas pesquisas em consolidação de MVs utilizam a migração como principal mecanismo para gerenciar eficientemente a infraestrutura da nuvem. Porém a sobrecarga gerada pelo processo de migração pode perturbar a qualidade da rede, causando problemas como de perda de pacotes, e redução da largura de banda disponível. Para quantificar esse impacto, é desenvolvida uma solução que realiza um conjunto automatizado de testes de migração, e é quantificada a qualidade da conexão com a MV através das aplicações *ping* e *iperf3*. Enquanto *ping* verifica a disponibilidade da MV, o *iperf* gera uma carga TCP para verificar o impacto na largura de banda. Por fim, são gerados relatórios de resumo da migração, com os valores de média, mediana, desvio padrão, máximo e mínimo com a aplicação *R*, podendo-se adicionalmente gerar gráficos.

Com base nos resultados obtido em uma suíte de testes, observa-se que a migração interfere na largura de banda da rede e nas falhas de retransmissão física da infraestrutura da nuvem. Deste modo, conclui-se que uma estratégia de consolidação que se baseia em altas taxas de consolidação, pode afetar a qualidade da rede e causar o efeito rebote na eficiência ao gerar outros problemas enquanto tenta solucionar a eficiência.

Referências

[Abdelsamea et al. 2017] Abdelsamea, A., El-Moursy, A. A., Hemayed, E. E., and Eldeeb, H. (2017). Virtual machine consolidation enhancement using hybrid regression algorithms. *Egyptian Informatics Journal*.

[Ahmad 2015] Ahmad, R. W. *et. al*. (2015). A survey on virtual machine migration and server consolidation frameworks for cloud data centers. *JNCA*, 52:11–25.

[Bala and Devanand 2016] Bala, M. and Devanand (2016). Virtual Machine Migration: A Green Computing Approach in Cloud Data Centers. In *Proceedings of the International Congress on Information and Communication Technology*, pages 161–168. Springer, Singapore. DOI: 10.1007/978-981-10-0755:218.

[Beloglazov 2013] Beloglazov, A. (2013). Energy-efficient management of virtual machines in data centers for cloud computing. *PHD Thesis, University Of Melbourne, AU*. Phd thesis.

[Beloglazov et al. 2012] Beloglazov, A., Abawajy, J., and Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Gener. Comput. Syst.*, 28(5):755–768.

[Beloglazov and Buyya 2012] Beloglazov, A. and Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency Computat.: Pract. Exper.*, 24(13):1397–1420.

[Deshpande et al. 2012] Deshpande, U., Kulkarni, U., and Gopalan, K. (2012). Inter-rack Live Migration of Multiple Virtual Machines. In *Proceedings of the 6th international workshop on Virtualization Technologies in Distributed Computing Date*, VTDC '12, pages 19–26, New York, NY, USA. ACM.

[Hameed et al. 2016] Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P. P., Kolodziej, J., Balaji, P., Zeadally, S., Malluhi, Q. M., Tziritas, N., Vishnu, A., Khan, S. U., and Zomaya, A. (2016). A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing*, 98(7):751–774.

[Medina and García 2014] Medina, V. and García, J. M. (2014). A Survey of Migration Mechanisms of Virtual Machines. *ACM Comput. Surv.*, 46(3):30:1–30:33.

[Mell and Grance 2011] Mell, P. M. and Grance, T. (2011). SP 800-145. The NIST Definition of Cloud Computing. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States.

[Mosa and Paton 2016] Mosa, A. and Paton, N. W. (2016). Optimizing VM placement for energy and SLA in clouds using utility functions. *Journal of Cloud Computing*, 5.

[Nathuji and Schwan 2007] Nathuji, R. and Schwan, K. (2007). VirtualPower: Coordinated Power Management in Virtualized Enterprise Systems. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 265–278, New York, NY, USA. ACM.

[Patel and Vaghela 2016] Patel, S. D. and Vaghela, D. (2016). A survey paper on load balancing algorithms for resource management in virtualization. *International Journal For Innovative Research In Science And Technology*, 2(11):68–70.

[Rohani and Roosta 2014] Rohani, H. and Roosta, A. K. (2014). Calculating Total System Availability.

[Xu et al. 2016] Xu, M., Tian, W., and Buyya, R. (2016). A survey on load balancing algorithms for vm placement in cloud computing. *Distributed, Parallel, And Cluster Computing (cs.dc)*.