Um SLA 3D para o Escalonamento Multiusuário Sobre Múltiplos Provedores de IaaS

Cristiano C. A. Vieira[1], **Luiz F. Bittencourt**[1], **Edmundo R. M. Madeira**[1]

[1]Universidade de Campinas - Instituto de Computação
Av. Albert Einstein, 1251 - Cidade Universitária, Campinas/SP - Brasil

cargemon@facom.ufms.br, bit@ic.unicamp.br, edmundo@ic.unicamp.br

Abstract. *Customers of Cloud providers run applications with different Quality of Service (QoS) requirements. However, the main item treated in SLAs is currently the time of use and the implications for their violation, neglecting other important factors in an SLA. To address this issue, in this paper, we introduce an SLA that considers three important aspects of QoS: reliability, performance and security. The goal is to schedule virtual machine requests from users to public IaaS providers in order to decrease the monetary cost without violating the requests QoS. We present an integer linear program (ILP) formulation and a heuristic to compute the scheduling. In addition, we discuss the experimental results obtained from the implementation of our strategy.*

Resumo. *Usuários de provedores de nuvens executam aplicações com diferentes necessidades de QoS. Contudo, o principal item tratado nos SLAs atualmente é o tempo de utilização e as implicações quanto a sua violação, negligenciando outros fatores importantes em um SLA. Para tratar esta questão, neste trabalho, introduzimos um SLA que considera três importantes aspectos de QoS: confiabilidade, desempenho e segurança. O objetivo é escalonar requisições de máquinas virtuais de múltiplos usuários em provedores públicos de IaaS obtendo o menor custo monetário sem violar a QoS das requisições. Apresentamos uma PLI e uma heurística para computar o escalonamento. Além disso, apresentamos e avaliamos os resultados experimentais obtidos a partir da implementação da nossa estratégia.*

1. Introdução

A computação em nuvem atrai cada vez mais usuários a consumirem e provedores a oferecerem serviços em nuvem por meio do modelo *pay-as-you-go*. Os usuários terceirizam suas demandas computacionais para os provedores públicos e pagam de acordo com sua utilização. Neste cenário, um crescente número de empresas está adotando serviços de computação em nuvem, tais como infraestrutura (IaaS), plataforma (PaaS) e software (SaaS) [Zhang et al. 2010]. Considerando serviços de infraestrutura, usuários alugam capacidade de processamento a partir de provedores de IaaS (Amazon EC2[1], FlaxiScale [2], Google Compute Engine[3], ElasticHosts[4], CloudSigma[5] e JoyentCloud[6]) sob a

[1]Elastic Cloud Computing - hwwps://aws.amazon.com/ec2
[2]FlexiScale Cloud Comp and Hosting - http://www.flaxiscale.com
[3]Google Compute Engine - https://cloud.google.com/compute/
[4]ElasticHosts - http://www.elastichosts.com/
[5]CloudSigma - https://www.cloudsigma.com/
[6]JoyentCloud - https://www.joyent.com/

forma de máquina virtual (VM - *Virtual Machine*) onde podem implantar suas aplicações sob diferentes modelos de cobrança, tais como OD (*On Demand*), RE (*Reserved*) e SP (*Spot*) [Vieira et al. 2015]. Instâncias sob o modelo OD possuem o maior custo. Contudo, o maior nível de confiabilidade. São interrompidas apenas pelo usuário. Instâncias RE possuem o custo menor comparado ao do OD. Contudo, o usuário precisa pagar uma taxa para reservar sua utilização por um período de tempo. Instâncias SP possuem o menor custo e também a menor confiabilidade, pois podem ser interrompidas pelos provedores caso o valor oferecido pelo usuário seja inferior ao valor cobrado pelo provedor.

Existem inúmeros objetivos complexos intrínsecos aos pontos de vista do usuário e do provedor na comercialização através do modelo *pay-as-you-go*. Sob o ponto de vista do usuário, o objetivo é beneficiar-se da computação em nuvem obtendo alta disponibilidade, redução de custos, balanceamento de carga e melhoria na tolerância a falhas. A seleção dos recursos depende das necessidades dos usuários e de suas aplicações.

Várias características em um SLA (*Service Level Agreement*) oferecido pelo provedor de IaaS devem ser levadas em consideração durante o escalonamento das VMs para o usuário executar suas aplicações, tais como: confiabilidade, desempenho, segurança, localização, tempo de execução e outras. Ao consideramos diferentes itens no SLA o usuário consumidor da nuvem encontra dificuldades em escolher o melhor conjunto de recursos para atender suas necessidades e reduzir os custos. Neste contexto, várias plataformas, como os *Brokers*, têm sido propostas para oferecer o serviço de descoberta e a alocação de recursos sobre múltiplos provedores de IaaS ajudando os usuários neste processo de decisão. Contudo, nenhuma considera o escalonamento de múltiplos usuários e diferentes características no SLA. Vieira et. al [Vieira et al. 2015] propuseram uma PaaS que oferece o serviço de escalonamento considerando duas características no SLA: confiabilidade e desempenho, e realiza o escalonamento de requisições de apenas um usuário.

Ao considerarmos vários usuários durante o escalonamento, a questão de segurança é evidente. Diferentes usuários requerem diferentes políticas de segurança para executar suas aplicações. Em geral, alcançar um nível satisfatório de segurança para serviços fornecidos em nuvem é um desafio dada a complexidade, heterogeneidade e natureza dinâmica das necessidades dos usuários e disponibilidade dos serviços oferecidos pelos provedores [Silva and de Geus 2015]. Neste trabalho, abordamos o problema de escalonamento de requisições de VMs de usuários distintos sobre recursos de múltiplos provedores de IaaS. Nossa contribuição é a extensão da PaaS apresentada em [Vieira et al. 2015] por tratar requisições de vários usuários e considerar a segurança no escalonamento. Para isso propomos neste trabalho: a) Um SLA com três parâmetros: confiabilidade, desempenho e segurança; b) Uma PLI para computar o escalonamento utilizando o SLA proposto; e c) Um algoritmo heurístico para evitar violação de QoS; e d) Avaliação dos resultados experimentais.

O restante do texto está organizado como a seguir: na próxima seção, descrevemos uma visão geral sobre os trabalhos relacionados ao tema. Em seguida, na Seção III, caracterizamos as necessidades dos usuários de provedores de IaaS e definimos o problema abordado. Na Seção IV, apresentamos nossa proposta de escalonamento multiusuário. Na Seção V, apresentamos os resultados. Na Seção VI apresentamos nossas conclusões.

2. Trabalhos Relacionados

Várias plataformas como *Brokers* foram propostas para otimizar a seleção de recursos e escalonar as requisições dos usuários sobre os provedores de IaaS. Em [Nair et al. 2010] são discutidas questões de segurança associadas com *cloud brokerage* e apresentada uma arquitetura de *framework* capaz de regular serviços em *cloud bursting*. Em [Tordsson et al. 2012] é apresentado um *broker* que otimiza a alocação de recursos sobre múltiplas nuvens. Similarmente, em [Leitner et al. 2012] é apresentada uma estratégia para computar o escalonamento de requisições sobre nuvens que tenta diminuir a violação do SLA. Em [Shen et al. 2013], é proposta uma estratégia híbrida de escalonamento que minimiza o custo do aluguel por fazer uso instâncias OD e RE. Uma PLI é formulada para realizar o escalonamento. Os três trabalhos não realizam o escalonamento multiusuário e nem tratam questões de segurança no SLA.

Considerando o aspecto de segurança, Marcon et al. [Marcon et al. 2013] introduziram uma estratégia de alocação que aumenta a segurança dos recursos de rede compartilhados entre várias aplicações. Eles agrupam requisições de usuários confiáveis na mesma infraestrutura virtual. Uma PLI computa o escalonamento onde segurança e isolamento são obtidos por alocar cada grupo sobre diferentes infraestruturas virtuais. Diferencia-se do nosso trabalho, pois consideramos que não existe comunicação entre as VMs.

Utilizando heurísticas, Assunção et al. [Assunção et al. 2010] investigaram sete estratégias de escalonamento que consideram o uso de recursos de provedores de IaaS para expandir a capacidade computacional da infraestrutura local para reduzir o tempo de respostas das requisições dos usuários. Similarmente, Le et al. [Le et al. 2013] propuseram uma política adaptativa de gerenciamento de recursos que trata o término de execução das aplicações. Genaud e Gossa [Genaud and Gossa 2011] descreveram doze estratégias baseadas em heurísticas para o escalonamento *on-line* com o objetivo de minimizar o tempo e o custo do escalonamento. Os autores simularam as estratégias sobre um grid.

Em [Casalicchio and Silvestri 2012] as características de disponibilidade, desempenho e segurança são tratadas durante o escalonamento, contudo, sob o ponto de vista do provedor que objetiva aumentar o lucro obtido ao alugar os recursos.

Vieira et al. [Vieira et al. 2015] propuseram uma PaaS que oferece o serviço de escalonamento de requisições de usuário sobre diferentes provedores de IaaS considerando um SLA com duas características: confiabilidade e desempenho. Contudo, trata requisições de apenas um usuário. Os trabalhos existentes na literatura consideram apenas algumas das necessidades de QoS dos usuários separadamente, não o fazem em um único SLA. Este trabalho objetiva preencher esta lacuna ao passo que propõe uma PaaS que considera três características das necessidades do usuário (confiabilidade, desempenho e segurança) em um SLA. Além disso, realiza o escalonamento de requisições pertencentes a vários usuários ao mesmo tempo objetivando o menor custo monetário.

3. Caracterização das Necessidades dos Usuários e a Formulação do Problema

Usuários da web têm utilizado os serviços de computação em nuvem para diminuir os custos monetários com infraestrutura. Os usuários solicitam VMs de provedores de IaaS para executar suas aplicações. Desta forma, definimos a solicitação de uma instância de

VM como *requisição de VM*. Um usuário pode solicitar várias VMs com diferentes níveis de requisitos de QoS. Por outro lado, os provedores utilizam SLAs para especificar as características dos serviços oferecidos em termos de métricas que devem ser aceitas pelas partes, e define as penalidades para as violações do acordo. Existem vários parâmetros importantes em um SLA administrado pelo provedor de IaaS, tais como: tempo, custo, confiabilidade, segurança, localização, desempenho, comunicação, e armazenamento.

Neste trabalho, consideramos três parâmetros no SLA: confiabilidade, desempenho e segurança. Assim, definimos a requisição de uma instância de VM como $r = \{d, d\alpha, qos_c, qos_d, qos_s\}$, onde $r(d)$ representa o total de tempo que a instância deve ficar disponível para o usuário; $r(d\alpha)$ representa um relaxamento sobre d; $r(qos_c)$, $r(qos_d)$ e $r(qos_s)$ representam, respectivamente, as necessidades de confiabilidade; desempenho; e segurança. Denotamos como $\mathcal{R}$ o conjunto de requisições de um usuário. Uma VM é definida como $\psi = \{qos_c, qos_d, c\}$, onde $\psi(qos_c)$ é a confiabilidade da VM; $\psi(qos_d)$ é o desempenho da VM; e c é o custo por alugar a VM por uma unidade de tempo. A PaaS gerencia um conjunto $|\Psi| = m$ de instâncias de VMs alugadas de provedores de IaaS. Descrevemos uma modelagem dos parâmetros confiabilidade e desempenho que deriva da proposta em [Vieira et al. 2015] como características da necessidade do usuário. Uma modelagem do parâmetro de segurança é proposta na Seção 4.

3.1. Modelagem dos Requisitos de Confiabilidade

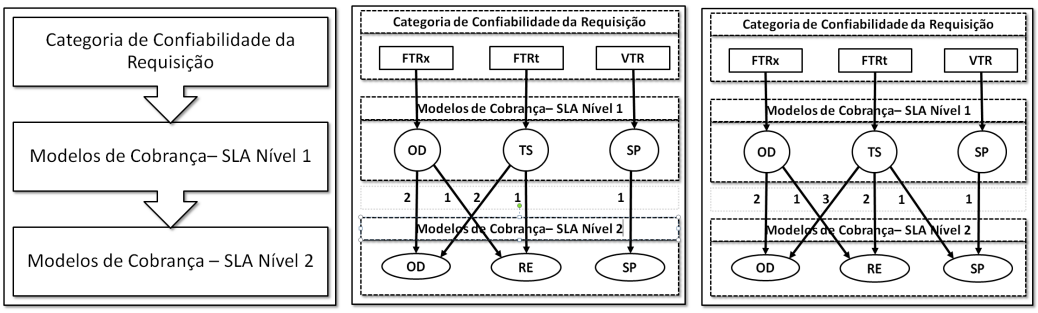
A confiabilidade define se um serviço pode sofrer atraso e/ou ser interrompido. Considerando o tempo de execução e a disponibilidade, classificamos o nível de confiabilidade de requisição do usuário em três categorias. Na primeira categoria, denominada Requisição de Tempo Fixo (FTRx), a requisição deve iniciar imediatamente e a VM não pode ser interrompida (preemptada) durante a execução. Na segunda, denominada Requisição de Tempo Flutuante (FTRt), a requisição pode não iniciar imediatamente. Contudo, uma vez iniciada não pode ser interrompida. E finalmente, na terceira categoria, denominada Requisição de Tempo Variável (VTR), a requisição pode não iniciar imediatamente e pode ser interrompida. Isto é, pode ser fragmentada em requisições menores. Por meio da classificação descrita é possível desenvolver um escalonamento que atenda com mais qualidade as necessidades do usuário referentes ao tempo de execução e à disponibilidade.

3.1.1. Modelo Conceitual e Duas Possibilidades de Mapeamento

Para a plataforma suportar a diversidade existente entre as necessidades do usuário e as características do serviço oferecido pelos provedores, propomos um modelo conceitual de interação entre estas informações (Fig. 1-a). O modelo conceitual possui dois níveis de SLA. O SLA de nível 1 corresponde ao SLA da plataforma e o SLA de nível 2 corresponde ao SLA pertencente aos provedores, respectivamente, denotados por Ω^1 e Ω^2. O nível de categorias de confiabilidade contém as três categorias de confiabilidade descritas anteriormente (FTRx, FTRt e VTR), as quais são mapeadas para algum modelo de cobrança no SLA de nível 1 ($\Omega^1 = \{OD, TS^7 \text{ e } SP\}$) pertencente à plataforma. Em

[7]Modelo *Time Slotted Reservation* (TS) . Neste modelo, o usuário solicita a utilização de uma VM em janelas de tempo [Vieira et al. 2014], e a execução não pode se interrompida. Isto permite utilizar melhores os recursos de instâncias RE, minimizando o curso por unidade de tempo.

seguida, é mapeado para um modelo de cobrança ($\Omega^2 = \{OD, RE \text{ e } SP\}$) de algum provedor público no SLA de nível 2. Desta forma temos a interação entre o usuário e a plataforma e em seguida, a interação entre a plataforma e os provedores de IaaS. Podemos incluir novas características na plataforma sem influenciar diretamente no SLA de nível 2 do modelo conceitual. Além disso, permite que novos modelos de cobrança sejam propostos sem interferir nos modelos atualmente disponíveis nos provedores de IaaS.



(a) (b) (c)

Figura 1. a) Modelo conceitual de mapeamento, b) Mapeamento Conservador, e c) Mapeamento Flexível (Adaptado de [Vieira et al. 2014])

A maneira com a qual o modelo conceitual foi proposta, em níveis, permite que uma série de mapeamentos seja possível. A Fig. 1(b) apresenta uma possibilidade de mapeamento denominada Mapeamento Conservador (MC). Sejam C_{od}, C_{re} e C_{sp}, respectivamente, o custo de uma instância, OD, RE e SP. Temos $C_{od} > C_{re} > C_{sp}$. Desta forma, a prioridade de utilização das instâncias tendem a ser SP, RE e OD, nesta ordem.

Outra possibilidade de mapeamento, denotada como Mapeamento Flexível (MF), Fig. 1(c), utiliza instâncias SP para diminuir o custo do escalonamento. Isso introduz uma potencial violação na QoS da requisição, pois o provedor pode interromper a VM SP. Para evitar isso, a plataforma gerencia duas zonas de VMs alugadas: Zona de Execução (ZE) e Zona de Redundância (ZR). A ZE contém as VMs OD, RE e SP pertencentes ao conjunto Ψ, enquanto que a ZR é composta por m' VM RE no conjunto Ψ'. Para cada requisição r, na qual $r(qos_r) = FTRt$, alocada em uma VM SP na ZE, é criada uma cópia de r denominada r' na ZR que será utilizada se r falhar. O grau de concorrência σ^c é a quantidade de requisições que podem executar concorrentemente em uma VM em Ψ'.

3.2. Modelagem dos Requisitos de Desempenho

Consideramos que o desempenho de uma requisição do usuário é a configuração de hardware necessária para executar a requisição durante um certo tempo de duração. Por outro lado, os provedores de IaaS oferecem diferentes VMs com diferentes características de desempenho. Seja ρ uma métrica de desempenho, $1\rho < 2\rho < ... < n\rho$. Tanto a requisição do usuário quanto a VM oferecida pelo provedor possuem o desempenho qos_d associado. Desta forma, mapeamos a requisição para a instância de VM utilizando a seguinte regra: $r \rightarrow \psi$, se $r(qos_p) = \psi(qos_p)$. Esta não é uma condição suficiente para que a requisição seja escalonada, pois a confiabilidade e a segurança devem ser consideradas também.

3.3. Formulação do Problema

Muitas plataformas têm sido propostas para computar o escalonamento e ajudar os usuários a escolherem os melhores recursos para executar suas aplicações. O nível de

QoS especificado pelo usuário na requisição deve ser considerado durante o processo de escalonamento. A PaaS proposta em [Vieira et al. 2015] inovou ao considerar os requisitos de confiabilidade e desempenho durante o escalonamento. Neste artigo, propomos a extensão da PaaS para incluir o requisito de segurança no SLA. Isso possibilita o escalonamento de requisições de vários usuários ao mesmo tempo. O usuário u_i submete um conjunto $\mathcal{R}_i$ de requisições para a PaaS. O número de usuários é denotado como n_u. Definimos como $\mathcal{R}^+ = \bigcup \mathcal{R}_i$, $1 \leq i \leq n_u$, o conjunto de todas as requisições. A partir disso, formulamos nosso problema como segue:

Compute um escalonamento S de um conjunto $\mathcal{R}^+$ de requisições de VMs sobre um conjunto Ψ de instâncias de VMs de provedores de IaaS objetivando o menor custo de alocação sem violar a QoS das requisições.

A PaaS computa o escalonamento $S = \{t; q; c; E\}$, de $\mathcal{R}^+$, onde: t representa o tempo total de execução de $\mathcal{R}^+$; q representa o número de requisições atendidas; c representa o menor custo para o usuário; e E representa o escalonamento contendo o tempo inicial de cada requisição alocada e sua respectiva VM. Por considerar vários usuários ao mesmo tempo durante o escalonamento, um novo problema é identificado: *Como tratar as características de segurança em um escalonamento de usuários distintos?*

Para contornar este problema propomos o isolamento e compartilhamento de requisições confiáveis e incluímos a segurança como um parâmetro no SLA. Uma estratégia ingênua de escalonamento pode elevar o custo monetário. Portanto, é importante produzir bons escalonamentos utilizando algoritmos mais elaborados. Alocação de recursos baseada em SLA com diferentes parâmetros em computação em nuvem é um problema NP-Difícil [Bittencourt et al. 2012].

4. Um Escalonamento Multiusuário Sobre Múltiplos Provedores de IaaS

Propomos uma modelagem do requisito de segurança para estender o SLA proposto em [Vieira et al. 2015] para um SLA tri-dimensional que considera a confiabilidade, o desempenho e a segurança como dimensões. Apresentamos um algoritmo para computar o escalonamento utilizando o SLA proposto.

4.1. Modelagem dos Requisitos de Segurança

Propomos a geração de grupos de isolamento e de compartilhamento de requisições entre usuários confiáveis. Apresentamos quatro possibilidades de geração de grupos (Fig. 2), descritas a seguir:

(a) Isolamento completo: Requisições entre usuários distintos não são executadas na mesma VM. Um grupo g_i é criado para as requisições $\mathcal{R}_i$ de cada usuário, $1 \leq i \leq n_u$;
(b) Isolamento parcial: Parte das requisições de usuários distintos são atribuídas a um mesmo grupo (g_0) e podem ser executadas na mesma VM. Contudo, uma outra parte das requisições continuam isoladas completamente;
(c) Compartilhamento entre grupos confiáveis: Um grupo para mais de um usuário, possibilitando o compartilhamento de grupos de requisições de usuários confiáveis.
(d) Compartilhamento completo: Uma requisição pode ser executada com qualquer outra. Apenas um grupo é criado utilizando este modelo de geração de grupos.

Outras possibilidades podem ser incluídas como mapeamento para que o escalonamento atenda às necessidades dos usuários da plataforma.

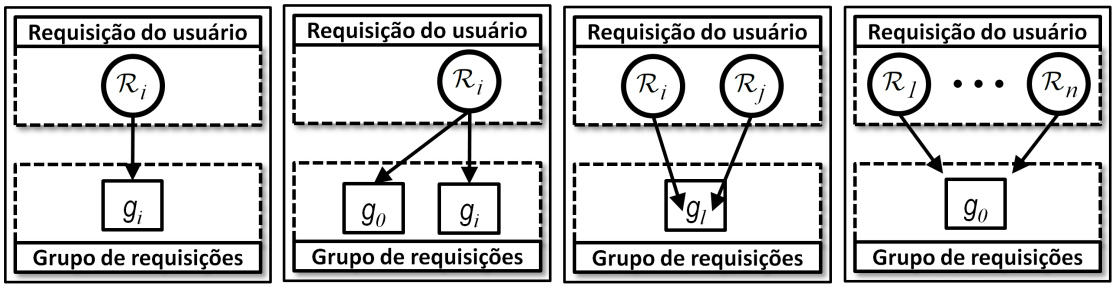


Figura 2. Possibilidades de modelos de geração de grupos

4.2. Um Algoritmo para Computar o Escalonamento

Propomos um algoritmo em quatro fases. O Algoritmo 1 apresenta a estratégia proposta.

Algoritmo 1 Escalonamento Multi-Usuário

Entrada: $\mathcal{R}^+$, Ψ, σ^c, m^g, m^c
Saída: S

1: $\mathcal{R}^+_{\mathcal{G}} \leftarrow$ *Geração dos grupos de isolamento*($\mathcal{R}^+$, m^g) {Primeira Fase}
2: $\mathcal{R}_{SLA^2} \leftarrow$ *Mapeamento das categorias de cofiabilidade*($\mathcal{R}^+_{\mathcal{G}}$, m^c) {Segunda Fase}
3: $S_{ze} \leftarrow$ *Computação da zona de execução*($\mathcal{R}_{SLA^2}$, Ψ) {Terceira Fase}
4: $S_{zr} \leftarrow$ *Computação da zona de redundância*(S_{ze}, σ^c) {Quarta Fase}
5: Compute $S \leftarrow S_{ze} + S_{zr}$

O modelo para geração de grupos de requisições m^g (Fig. 2) é utilizado para geração dos grupos (primeira fase). O modelo de mapeamento das categorias de confiabilidade m^c é utilizado para o mapeamento das categorias (segunda fase). Utilizamos uma PLI para computar a zona de execução (terceira fase) e uma heurística baseada no grau de utilização das VMs para computar a zona de redundância (quarta fase).

4.3. Primeira Fase: Geração dos Grupos de Isolamento

Realizamos o agrupamento de requisições de usuários confiáveis em grupos do conjunto $\mathcal{G} = \{g_0, g_1, g_2, ..., g_{n_g}\}$. Denotamos n_g como o número de grupos criados. A primeira fase do Algoritmo 1 é computada utilizando um dos modelos apresentados na Fig. 2.

4.4. Segunda Fase: Mapeamento das Categorias de Confiabilidade

Durante a segunda fase, realizamos o mapeamento das requisições dos usuários seguindo o modelo especificado em m^c. Atualmente, a plataforma disponibiliza dois tipos de mapeamento como descrito anteriormente: mapeamento conservador e mapeamento flexível.

4.5. Terceira Fase: Computação da Zona de Execução

Para computar o escalonamento da ZE, incluindo os grupos de isolamento e de compartilhamento de requisições, introduzimos duas importantes modificações na PLI proposta em [Vieira et al. 2015]: a primeira está relacionada à inclusão da terceira dimensão no SLA utilizando a geração de grupos computados na primeira fase do Algoritmo 1, a segunda modificação está relacionada a permitir mais de uma requisição ser executada ao

mesmo tempo na mesma VM. A PLI computa o escalonamento na terceira fase utilizando as variáveis binárias u, v, w, x, y, e z e as constantes C, $\mathcal{B}$ e $\mathcal{K}$ como a seguir:

- $u_{t,\psi}$: Assume o valor 1 se ψ executa alguma requisição no instante t. Senão, 0;
- $v_{g,\psi}$: Assume o valor 1 se ψ executa alguma requisição do grupo g. Senão, 0;
- w_r: Assume o valor 1 se a requisição r é executada. Senão, 0;
- $x_{r,\psi}$: Assume o valor 1 se r é executada em ψ. Senão, 0;
- $y_{r,t,\psi}$: Assume o valor 1 se r é executada em ψ no instante t. Senão, 0;
- $z_{r,t,\psi}$: Assume o valor 1 se a requisição r com $r(qos_c) = \{FTRt\}$ inicia a execução no instante t sobre a VM ψ. Senão, 0;
- $C_{t,\psi}$: Constante que assume o custo por unidade de tempo por utilizarmos a VM.
- $\mathcal{B}$: Constante suficientemente grande que assume um peso para cada requisição.
- $\mathcal{K}$: Constante suficientemente grande utilizada para garantir que requisições FTRt iniciem apenas uma vez.

Formulamos a função objetivo $F = \sum_{t \in T} \sum_{\psi \in \Psi} (u_{t,\psi} \times C_{t,\psi}) - \sum_{r \in \mathcal{R}} (w_r \times \mathcal{B})$ que computa o escalonamento de $\mathcal{R}^+$ sobre Ψ objetivando o menor custo de alocação sem violar o QoS da requisição. Devemos minimizar F sujeito às seguintes restrições:

$$\sum_{t=1}^{r(d\alpha)} y_{r,t,\psi} = r(d) \times x_{r,\psi}; \forall r \in \mathcal{R}, \forall \psi \in \Psi \quad (1)$$

$$\sum_{g \in G} v_{g,\psi} \leq 1; \forall \psi \in \Psi \quad (2)$$

$$x_{r,\psi} \leq v_{r(g),\psi}; \forall \psi \in \Psi, \forall r \in \mathcal{R} \quad (3)$$

$$\sum_{\psi \in \Psi} x_{r,\psi} = w_r; \forall r \in \mathcal{R} \quad (4)$$

$$\sum_{r \in \mathcal{R}} (y_{r,t,\psi} \times r(qos_d)) = u_{t,\psi} \times \psi(k); \forall t \in \left[1, r(d\alpha)\right], \forall \psi \in \Psi \quad (5)$$

$$\sum_{r \in R} y_{r,t,\psi} = 0; \forall t \in T; \psi \in \Psi; MF(r, \psi) = 0 \quad (6)$$

$$v_{g,\psi}, w_r, x_{r,\psi}, y_{r,t,\psi}, z_{r,t,\psi} \in \{0, 1\}; \forall g \in G, \forall r \in \mathcal{R}, \forall t \in \mathcal{T}, \forall \psi \in \Psi \quad (7)$$

$$\sum_{t=1}^{\mathcal{L}} z_{r,t,\psi} = x_{r,\psi}; \forall r \in \mathcal{R}, \forall \psi \in (\Psi^{od} \bigcup \Psi^{re}) \quad (8)$$

$$r(d) - \mathcal{K} \times (1 - z_{r,t,\psi}) \leq \sum_{s=t}^{r(d)+t-1} y_{r,s,\psi} \leq r(d) + \mathcal{K} \times (1 - z_{r,t,\psi}) \quad (9)$$

$$\forall r \in \mathcal{R}, \forall t \in \left[1, r(d\alpha) - r(d) + 1\right], \forall \psi \in \Psi, r(qos_c) = \{FTRx, FTRt\}$$

A restrição (1) especifica que se uma requisição for executada, ela deve ser executada em $r(d)$ unidades de tempo e em uma única VM. As restrições (2) e (3) trabalham juntas e especificam que as requisições escalonadas em uma VM devem pertencer

ao mesmo grupo de isolamento. A primeira especifica que uma VM ψ deve atender requisições de no máximo um grupo. A segunda especifica que se a requisição r executar na VM ψ, a VM ψ deve executar somente requisições do grupo da requisição de r, denotado como $r(g)$. A restrição (4) especifica que se uma requisição for escalonada, ela contribuirá para o custo total.

A restrição (5) especifica que a soma do desempenho das requisições executadas em uma unidade de tempo deve ser igual ao desempenho da instância de VM na qual as requisições foram alocadas. A garantia de execução conforme o modelo de mapeamento entre SLA de nível 1 e SLA de nível 2 apresentado na Fig. 1 é feita pela restrição 6) em conjunto com a função de mapeamento FM(). A restrição (7) assegura que as variáveis u, v, w, x, y e z sejam binárias. As restrições (8) e (9) asseguram que requisições do tipo FTRt sejam executadas de maneira atômica.

A PLI computa o melhor escalonamento com o maior número de requisições atendidas. Uma requisição não pode ser atendida parcialmente. O valor da constante $\mathcal{B}$ é atribuído para cada requisição w_r atendida. Seja C_p o custo computado pela PLI. Então, $S.q = \lfloor \frac{-C_p}{\mathcal{B}} \rfloor + 1$ e $S.c = (S.q \times \mathcal{B}) - C_p$. Seja d_m e ψ_m, respectivamente, o maior tempo de execução entre todas as requisições $r \in \mathcal{R}^+$ e o maior custo por unidade de tempo entre todas as VMs $\psi \in \Psi$. Deste modo, $S.q$ e $S.t$ podem ser computados como descrito anteriormente somente se $\mathcal{B} > (d_m \times \psi_m)$. Esta é a maneira como $\mathcal{B}$ deve ser definida. Caso contrário, o resultado da PLI pode ser comprometido.

4.6. Quarta Fase: Computação da Zona de Redundância

Quando a terceira fase termina, o escalonamento computado pode ter gerado resultados onde requisições FTRt são escalonadas em VMs SP. Como descrito anteriormente, requisições FTRt não podem ser preemptadas e podem sofrer um atraso inicial estabelecido por $r(d\alpha)$. Contudo, VMs SP podem ser preemptadas e deste modo precisamos alocar estas requisições para VMs na zona de redundância, então elas podem ser ativadas no caso das VMs SP serem preemptadas na ZE. Definimos $\Psi^\circ = \{\psi_1, \psi_2, ..., \psi_{m^\circ}\}$ como o conjunto de VMs nas quais ao menos uma requisição FTRt tenha sido alocada. Quando a VM pertence ao grupo de isolamento g, utilizamos Ψ°_g. Então, na quarta fase do Algoritmo 1 devemos computar a ZR para as requisições em Ψ°.

Em [Vieira et al. 2014], propusemos o escalonamento Baseado no Grau de Utilização (BGU) para computar a ZR considerando o grau de utilização da VM no escalonamento da ZE. Neste trabalho apresentamos o Algoritmo 2, denominado *Grau de Utilização 3D* (GU-3D), para computar a ZR. O algoritmo é baseado no Algoritmo BGU, contudo, considera 3 dimensões no SLA: confiabilidade, desempenho e segurança.

Definimos o *grau de utilização* σ^u_ψ como a quantidade de unidades de tempo que uma VM ψ executa uma requisição de acordo com o escalonamento S_{ze} da zona de execução. Por exemplo, suponha que somente as requisições r_1 e r_2 do tipo FTRt com $r_1(d) = 25$ e $r_2(d) = 40$ foram escalonadas, r_1 antes de r_2, para executar na VM ψ_i em S_{ez}. Neste caso, $\sigma^u_{\psi_i} = 65$. A ideia básica do Algoritmo 2 é computar $\sigma^u_\psi, \forall\ \psi \in \Psi^\circ_g$ e então utilizar a ordem dada por σ^u para escalonar ψ para ψ', cada uma contendo σ^c VMs. Isto deverá ser feito de acordo com o modelo de geração de grupos especificado.

Na linha 1 computamos Ψ°_g, $g \in \mathcal{G}$, os grupos de isolamento de acordo com o modelo de grupos definido em m^g. O grau de utilização σ^u para cada VM ψ pertencente

Algoritmo 2 Grau de Utilização 3D (GU-3D)

Entrada: S_{ze}, σ^c, m^g
Saída: Schedule S_{rz}

1: **para cada** $g \in \mathcal{G}$ **faça**
2: $\quad \Psi_g^\circ \leftarrow$ IGG(Ψ°, m^g);{Compute os grupos de isolamento}
3: **final para**
4: **para cada** $g \in \mathcal{G}$ **faça**
5: $\quad$ **para cada** $\psi \in \Psi_g^\circ$ **faça**
6: $\quad\quad$ Compute σ_ψ^u;{Compute o grau de utilização de S_{ze}}
7: $\quad$ **final para**
8: **final para**
9: **para cada** $g \in \mathcal{G}$ **faça**
10: $\quad \Psi_g^\circ \leftarrow$ Orderne Ψ_g° por σ^u
11: $\quad m_g \leftarrow |\Psi_g^\circ|$
12: $\quad$ **para** $i = 1$ até m_g **faça**
13: $\quad\quad \psi'_{\lceil \frac{i}{\sigma^c} \rceil} \leftarrow \psi'_{\lceil \frac{i}{\sigma^c} \rceil} \bigcup \psi_g^i$ {Compute o escalonamento}
14: $\quad$ **final para**
15: **final para**
16: $S_{zr}.c$;{Compute o custo do escalonamento}
17: retorne S_{zr}

a cada grupo de isolamento é computado na linha 4. Então, ordenamos o conjunto Ψ_g° por σ^u para cada grupo de isolamento na linha 10 e computamos o escalonamento na linha 12 utilizando a ordenação. O custo monetário do escalonamento é dado pela soma das unidades de tempo na qual alguma VM ψ' tem uma requisição alocada para ela, multiplicada pelo custo de cada unidade de tempo. Então, na linha 16, computamos o custo do escalonamento $S_{rz}.c$. Denotamos $\psi'.t$ como uma unidade de tempo de uma VM que assume o valor 1 se ψ_i está executando uma requisição no tempo t. Assume o valor zero caso contrário. Deste modo, o valor de $S_{rz}.c$ é computado como: $S_{rz}.c = \sum \psi'.t \times C_{re}; \quad \forall\ \psi' \in \Psi', \quad 0 < t \leq T$.

A complexidade assintótica do Algoritmo 2 é dominada pela computação do grau de utilização σ^u e pela ordenação do conjunto Ψ°, resultando em $O(m^\circ \times T + m^\circ \log m^\circ)$, onde T é o tempo total considerado para a disponibilidade das VMs. Embora tenha uma complexidade polinomial, utilizando uma heurística, ele não garante a qualidade da solução. Isso acontece pelo fato de que ele considera apenas a quantidade de unidades de tempo utilizadas em uma VM, e não o instante de utilização.

Se uma preempção ocorrer, a concorrência na zona de redundância é imediatamente interrompida, deixando a cópia da requisição que foi preemptada na zona de execução concluir sua execução na zona de redundância sozinha permitindo que ela termine dentro do prazo do relaxamento $r(d\alpha)$ sem violar a QoS. Observe que a preempção de duas ou mais requisições na zona de execução que compartilham a mesma VM na zona de redundância causará falha à segunda (ou as posteriores) requisição preemptada, pelo fato de já ter sido excluída da zona de redundância. Denotamos C_1 e C_2, respectivamente, como os custos dos escalonamentos computados na terceira e na quarta fase. Além disso, denotamos C_3 como o custo para tratar a preempção das VMs durante a execução da

requisição. O custo C_3 é computado como $C_3 = \tau_{eer} \times C_{re} - \tau_{dp} \times C_{sp}$, onde, C_{re} e C_{sp} denotam o custo de utilização por unidade de tempo de uma instância de VM, respectivamente, do tipo RE na zona de redundância e do tipo SP na zona de execução. É necessário computar um tempo extra τ_{eer} e um desconto τ_{dp} pela VM interrompida na zona de execução, pois o custo já havia sido computado durante a terceira fase.

Denotamos o custo total do escalonamento C_t como $C_t = C_1 + C_2 + C_3$. Se não ocorrem preempções, temos $C_3 = 0$. Para que o escalonamento tenha um resultado satisfatório para o usuário, é importante que ele adéque inteligentemente às necessidades dos usuários. O SLA com três dimensões proposto contribui para este objetivo.

5. Resultados

Implementamos em Java o Algoritmo 1 para computar o escalonamento. As métricas observadas são o custo do escalonamento, a quantidade de requisições atendidas e a quantidade de VMs de cada tipo (OD, RE e SP) utilizadas. Utilizamos o IBM ILOG CPLEX com o limite de 90 minutos para cada execução da PLI. Realizamos 10 experimentos, E_i, $1 \leq i \leq 10$, cada um com 30 conjuntos de requisições, $E_i = \{R_1, R_2, ..., R_{30}\}$. Em cada experimento, o número de requisições em cada conjunto é E_i: $|R_j| = i \times 10$, $1 \leq j \leq 30$. Além disso, configuramos $r(d) \leftarrow$ *random*(1,20) e $r(d\alpha) \leftarrow$ *random*(1, 80)+$r(d)$ de cada requisição em cada conjunto $R_j \in E_i$. Como apresentado na Tabela 1 temos recursos disponíveis de dois provedores. O custo de cada VM é proporcional, multiplicado por 100, ao custo utilizado pela Amazon EC2 atualmente. O conjunto $\mathcal{R}$ é composto de requisições na proporção de de 20% (FTRx), 60%(FTRt) e 20%(VTR). Avaliamos diferentes graus de concorrência $\sigma^c = \{2, 4, 8\}$ na zona de redundância. Além disso, todas as requisições necessitam de apenas uma unidade de processamento (1ρ).

Tabela 1. Conjunto de VMs disponíveis para a plataforma.

Provedor	Desempenho	OD	$	RE	$	SP	$	TOTAL
1	1ρ	1	70.00	-	-	-	-	2
	2ρ	1	140.00	-	-	-	-	
2	1ρ	-	-	1	40.00	6	8,00	14
	2ρ	-	-	1	100.00	6	16,00	
							Total	**16**

Utilizamos a PLI e o Algoritmo 2 para computar, respectivamente, a zona de execução e a zona de redundância durante a execução dos 10 experimentos. Os resultados evidenciam o potencial da utilização das VMs do tipo SP no escalonamento. As Figs. 3(a), 3(b), 3(c) e 3(d) apresentam o custo do escalonamento obtido utilizando os mapeamentos conservador e flexível considerando, respectivamente, $n_u = 1$, $n_u = 2$, $n_u = 3$, e $n_u = 4$. A curva denotada por MC apresenta os resultados obtidos pela execução dos experimentos utilizando o MC. A curva denotada por ZE apresenta os resultados do escalonamento da zona de execução utilizando o MF.

Analisando os dados, observamos que para $n_u = 1$ (Fig. 3-a), a estratégia de utilizar o mapeamento flexível com $\sigma^c = 2$ apresenta um custo mais baixo comparado ao custo obtido pelo mapeamento conservador, exceto no Experimento E_{10}. Pois, neste experimento, temos muitas requisições e o custo de computar a zona de redundância é alto. Um comportamento semelhante pode ser observado para $n_u = 2$, (Fig. 3-b). Contudo, não

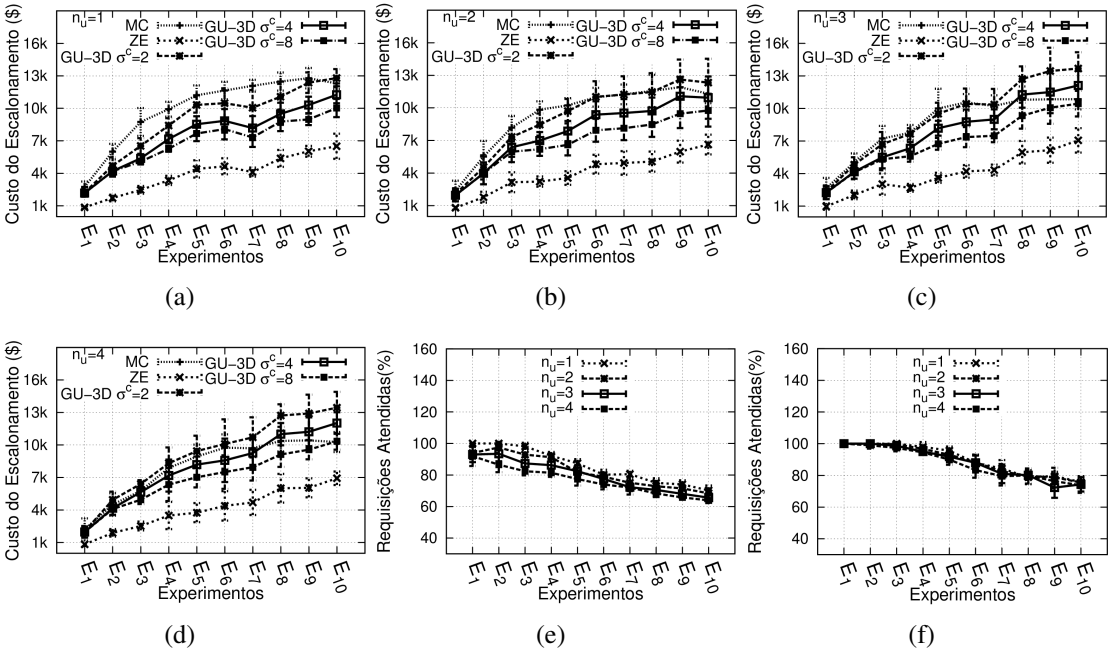


Figura 3. Custo do escalonamento e quantidade de requisições atendidas com $n_u = \{1, 2, 3, 4\}$.

é vantagem utilizar o mapeamento flexível com $\sigma^c = 2$ nos experimentos E_6, E_7, E_8, E_9 e E_{10}. No cenário com quatro usuários (Fig. 3-d), todos os experimentos com $\sigma^c = 8$ obtiveram um custo menor comparado ao obtido pelo escalonamento utilizando o mapeamento conservador. Por outro lado, utilizando $\sigma^c = 2$ todos os experimentos tiveram o escalonamento com custo superior comparado ao obtido pelo escalonamento conservador.

As Figs. 3(e) e 3(f) apresentam a quantidade de requisições atendidas utilizando, respectivamente, os modelos de mapeamento conservador e flexível. Podemos observar que, embora o escalonamento para apenas um usuário (n_1) tenha um custo maior, são atendidas mais requisições, quando comparado aos resultados do escalonamento para n_2, n_3 e n_4. Isso justifica a custo mais alto. Para $n_u = 4$, temos o menor custo. Contudo, a quantidade de requisições atendidas é menor.

A Fig. 4 apresenta a quantidade de VMs de cada tipo considerando os mapeamentos conservador em a), b) e c) e flexível em d), e) e f). Nos Experimentos E_1, E_2, E_3 e E_4, verificamos uma diferença quando comparamos a quantidade de VMs OD utilizada no mapeamento conservador (Fig. 4-a) e no mapeamento flexível(Fig. 4-d) . O mapeamento flexível utiliza menos requisições OD. Para o Experimento E_1, foram utilizadas aproximadamente 10% das VMs no mapeamento flexível e 41% no mapeamento conservador para $n_u = 1$. Esta diferença tem um impacto no custo, pois as VMs OD têm o maior custo. A quantidade de VMs do tipo RE é semelhante nos dois mapeamentos, Fig. 4(b) e Fig. 4(e), atingindo 100% de utilização no Experimento E_2 em diante. Por outro lado, a quantidade de VMs do tipo SP utilizada no mapeamento conservador, Fig. 4(c), é inferior à utilizada no mapeamento flexível, Figura 4(f). Desta forma, observamos que o mapeamento conservador utiliza mais VMs do tipo OD do que o mapeamento flexível; e menos VMs do tipo SP do que o mapeamento flexível. Este comportamento contribui para que o escalonamento computado utilizando o mapeamento flexível tenha um custo menor.

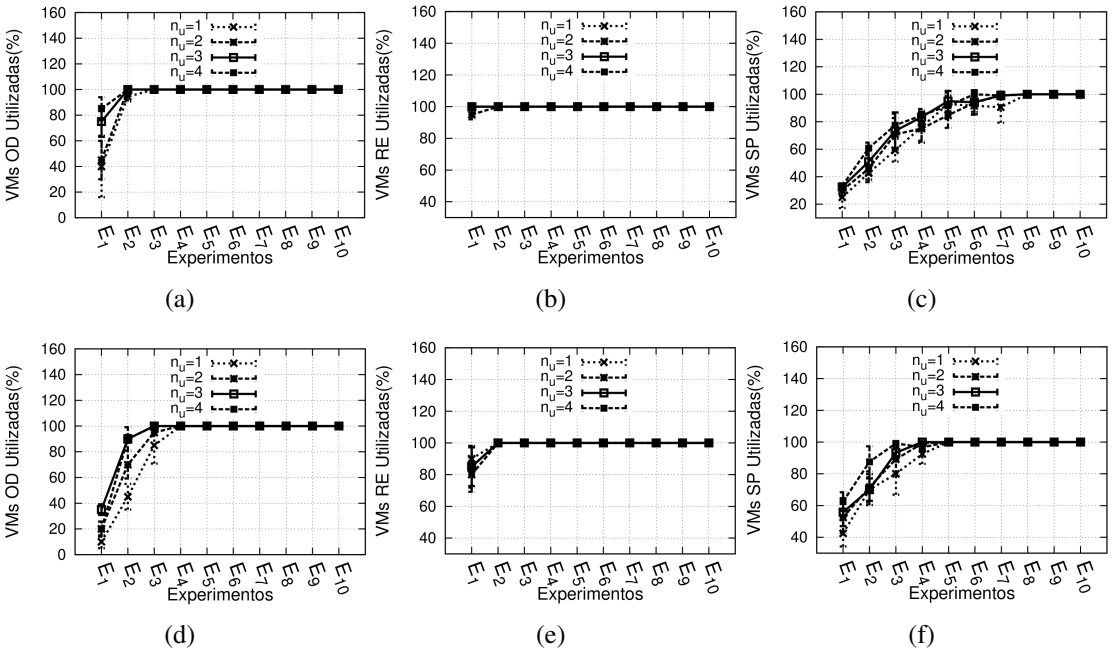


Figura 4. Quantidade de VMs utilizadas nos Mapeamento Conservador e Flexível

Observamos nos resultados dos experimentos realizados que quando consideramos requisições de usuários distintos e incluímos os grupos de isolamento, o custo do escalonamento aumenta. Isso ocorre pelo fato de termos que utilizar mais VMs para atender requisições em diferentes grupos que não podem compartilhar recursos. Esta é uma consequência por aumentar a segurança entre os usuários da PaaS.

6. Conclusão e Trabalhos Futuros

A grande diversidade de serviços e de provedores de IaaS dificulta a escolha dos melhores recursos para os consumidores implantarem suas aplicações. Considerar as características das necessidades do usuário é fundamental para escolher os serviços de maneira eficiente e reduzir os custos. Diante disso, apresentamos uma estratégia de escalonamento que utiliza um SLA com os parâmetros: confiabilidade, desempenho e segurança. Utilizando este SLA é possível considerar requisições de vários usuários no escalonamento além de se adequar melhor às necessidades do usuário enquanto diminui o custo do escalonamento e evita a violação de QoS. Os resultados experimentais mostram que ao introduzir a segurança no SLA, o custo do escalonamento pode aumentar. Contudo, possibilita que diferentes usuários tenham diferentes políticas de segurança. Como trabalho futuro, pretendemos introduzir o escalonamento *on-line* de requisições de diferentes usuários utilizando este SLA possibilitando que as requisições sejam modificadas ao longo do tempo.

Agradecimentos

Este trabalho foi parcialmente financiado pela CAPES e pelo CNPQ.

Referências

Assunção, M. D., Costanzo, A., and Buyya, R. (2010). A cost-benefit analysis of using cloud computing to extend the capacity of clusters. *Cluster Computing*, 13(3):335–347.

Bittencourt, L. F., Madeira, E. R. M., and da Fonseca, N. L. S. (2012). Scheduling in hybrid clouds. *Communications Magazine, IEEE*, 50(9):42–47.

Casalicchio, E. and Silvestri, L. (2012). An inter-cloud outsourcing model to scale performance, availability and security. In *Proceedings of 5th IEEE/ACM International Conference on Utility and Cloud Computing*, UCC 2012, Chicago, USA.

Genaud, S. and Gossa, J. (2011). Cost-wait trade-offs in client-side resource provisioning with elastic clouds. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 1–8.

Le, G., Xu, K., and Song, J. (2013). Dynamic resource provisioning and scheduling with deadline constraint in elastic cloud. In *Service Sciences (ICSS), 2013 International Conference on*, pages 113–117.

Leitner, P., Hummer, W., Satzger, B., Inzinger, C., and Dustdar, S. (2012). Cost-efficient and application sla-aware client side request scheduling in an infrastructure-as-a-service cloud. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 213–220.

Marcon, D., Oliveira, R., Neves, M., Buriol, L., Gaspary, L., and Barcellos, M. (2013). Trust-based grouping for cloud datacenters: Improving security in shared infrastructures. In *IFIP Networking Conference, 2013*, pages 1–9.

Nair, S., Porwal, S., Dimitrakos, T., Ferrer, A., Tordsson, J., Sharif, T., Sheridan, C., Rajarajan, M., and Khan, A. (2010). Towards secure cloud bursting, brokerage and aggregation. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 189–196.

Shen, S., Deng, K., Iosup, A., and Epema, D. (2013). Scheduling jobs in the cloud using on-demand and reserved instances. In *Proceedings of the 19th International Conference on Parallel Processing*, Euro-Par'13, pages 242–254, Berlin, Heidelberg. Springer-Verlag.

Silva, C. A. D. and de Geus, P. L. (2015). Return on security investment for cloud computing: a customer perspective. In *The 7th International Conference on Management of computational and collective IntElligence in Digital EcoSystems (MEDES'15)*, Caraguatatuba-SP, Brazil.

Tordsson, J., Montero, R. S., Moreno-Vozmediano, R., and Llorente, I. M. (2012). Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future Generation Computer Systems*, 28(2):358 – 367.

Vieira, C. C. A., Bittencourt, L. F., and Madeira, E. R. M. (2014). Reducing costs in cloud application execution using redundancy-based scheduling. In *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, pages 117–126.

Vieira, C. C. A., Bittencourt, L. F., and Madeira, E. R. M. (2015). A two-dimensional sla for services scheduling in multiple iaas cloud providers. *Int. J. Distrib. Syst. Technol.*, 6(4):45–64.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1:7–18.