Um Mecanismo para Compartilhamento de Recursos em Nuvens Colaborativas Baseado na Credibilidade dos Usuários

Hugo Sadok, Miguel Elias M. Campista e Luís Henrique M. K. Costa *

[1]Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA - DEL/POLI

{sadok, miguel, luish}@gta.ufrj.br

Resumo. *As nuvens colaborativas permitem que usuários com poucos recursos obtenham as mesmas vantagens de elasticidade das nuvens convencionais, ao contribuir com seus recursos ociosos. No entanto, quando a demanda por recursos é maior que a oferta, conflitos surgem e é necessário definir a alocação de cada usuário. Este trabalho propõe um mecanismo de alocação que garante a eficiência de utilização dos recursos ao mesmo tempo em que incentiva a colaboração. As propriedades do mecanismo proposto são demonstradas teoricamente, utilizando teoria dos jogos, a partir da definição de um modelo baseado em um cenário Bayesiano. Neste cenário os usuários constantemente decidem a quantidade de recursos que irão fornecer, ou requisitar, da nuvem colaborativa. Além disso, o mecanismo é verificado com simulações conduzidas através de traços reais da utilização da nuvem do Google. Os resultados mostram um aumento no número de requisições atendidas de até aproximadamente três vezes em comparação com a não utilização do mecanismo.*

Abstract. *Collaborative clouds bring the elasticity advantages from traditional clouds to users with few resources, as long as they contribute with their idle resources. Nevertheless, when demand for resources exceeds supply, conflicts arise and each user allocation must be defined. In this work, we propose an allocation mechanism that guarantees resource utilization efficiency while encouraging collaboration. Using game theory, we demonstrate the mechanism properties by defining a model based on a Bayesian setting. In such setting, users constantly decide the amount of resources to either contribute, or request, from the collaborative cloud. Moreover, we use simulations with the Google cloud dataset to verify the mechanism. The results show an increase in the number of served requests up to approximately three times compared to not using the mechanism.*

1. Introdução

O paradigma de computação em nuvem se baseia no provisionamento de recursos computacionais em centros de dados acessíveis pela rede. Além das vantagens trazidas pela terceirização da operação da infraestrutura, a nuvem fornece aos usuários maior flexibilização do dimensionamento da infraestrutura necessária às suas aplicações [Costa et al., 2012]. Sem a nuvem, os usuários precisam dimensionar seus recursos para atender os possíveis picos de demanda, causando ociosidade na maior parte

*Este trabalho foi parcialmente financiado pela CAPES, CNPq, FAPERJ e FINEP.

do tempo. Além disso, os provedores de nuvem conseguem oferecer serviços a preços competitivos por se beneficiarem da agregação e heterogeneidade de uso de vários usuários [Armbrust et al., 2010]. Entretanto, mesmo com o cenário de crescente popularização da computação em nuvem, muitos usuários (empresas, laboratórios, etc.) possuem sua própria infraestrutura computacional, seja por possuir uma base já instalada, seja por uma política de aquisição de equipamentos da instituição. A computação em nuvem neste caso se torna uma forma desses usuários obterem mais recursos quando necessário, e ao mesmo tempo compartilharem recursos de forma eficiente.

Algumas arquiteturas em nuvem são utilizadas para prover elasticidade, dentre elas a nuvem híbrida [Sotomayor et al., 2009]. Nuvens híbridas integram a infraestrutura local a uma nuvem pública, de forma que os usuários possam obter recursos adicionais em casos de pico de demanda. Uma pesquisa feita em 2016 com 1060 profissionais de TI mostrou que 71% usavam uma nuvem híbrida [Weins, 2016]. Embora os usuários possam obter recursos extras em caso de pico, é comum a presença de recursos ociosos. Uma alternativa à nuvem híbrida é o paradigma de nuvem colaborativa. Uma nuvem colaborativa é formada por máquinas dedicadas, cedidas pelos próprios usuários da nuvem. Esses usuários frequentemente estão geograficamente separados, dando um aspecto distribuído à nuvem. Os integrantes possuem acesso aos recursos ociosos dos demais, em seus momentos de pico, e contribuem com seus próprios recursos, quando ociosos. A nuvem colaborativa traz os benefícios da agregação e da heterogeneidade de uso, características das nuvens tradicionais, ao mesmo tempo em que permite que os usuários usem sua própria infraestrutura, como nas nuvens híbridas. Diferente do conceito de grades de computação, em que os usuários combinam recursos com um objetivo comum [Vaquero et al., 2008], os usuários da nuvem colaborativa podem ter objetivos distintos. Um exemplo de nuvem colaborativa é o Projeto GT-PID [Couto et al., 2015].

Embora as vantagens da agregação de recursos na nuvem colaborativa sejam facilmente vislumbradas, as decisões de alocação de recursos entre os usuários não são. No cenário onde os recursos de todos os usuários estão agregados em uma nuvem colaborativa, surgem as questões: Em caso de sobre-demanda, quais usuários devem receber mais recursos? Como garantir o interesse dos usuários em disponibilizar seus recursos ociosos? Este trabalho propõe uma solução baseada em princípios da teoria dos jogos. A solução envolve um modelo onde, a cada iteração, os usuários decidem a quantidade de recursos que desejam contribuir ou receber. Com base neste modelo, foi criado um mecanismo para gerir as alocações. Quando há mais recursos sendo ofertados do que pedidos, todos os pedidos são atendidos. Já quando os pedidos superam os recursos ofertados, as alocações são feitas baseadas nas credibilidades dos usuários, as quais são calculadas a partir das contribuições passadas. Este trabalho apresenta as seguintes contribuições, confirmadas a partir de demonstrações teóricas e de traços reais da nuvem do Google:

- **Um modelo de satisfação dos usuários de uma nuvem colaborativa.** O modelo define a satisfação em relação à quantidade de recursos recebidos de acordo com as necessidades dos usuários. O modelo também simplifica o projeto de mecanismos ao abstrair os recursos do usuário na nuvem e focar somente na quantidade de recursos ociosos e nos desejos dos usuários.
- **Um mecanismo de colaboração de recursos entre os usuários da nuvem.** O mecanismo garante que os usuários com maior contribuição no passado obtêm

mais recursos em caso de conflito, o que também cria um incentivo à colaboração. Mesmo com o uso da contribuição passada, o mecanismo possibilita a entrada de novos usuários. Além disso o mecanismo incentiva a verdade, de modo que os usuários não obtêm satisfação melhor ao mentir sobre suas necessidades.
- **Um algoritmo que rege as decisões de alocação do mecanismo.** O algoritmo se baseia nas restrições impostas pelo mecanismo para realizar a troca de recursos com base nas credibilidades dos usuários.

Este artigo está organizado da seguinte forma. A Seção 2 mostra uma visão geral do mecanismo e seus requisitos. A Seção 3 introduz o modelo de compartilhamento de recursos entre os usuários da nuvem. A Seção 4 propõe o mecanismo de alocações o qual é avaliado nas Seções 5 e 6. A Seção 7 descreve os trabalhos relacionados. Finalmente, a Seção 8 conclui o trabalho e aponta direções futuras.

2. Visão Geral do Mecanismo de Alocação

Nas nuvens tradicionais, os provedores se beneficiam da chamada multiplexação estatística, em que a necessidade por recursos agregada de diversos usuários evita que o provedor tenha que dimensionar a nuvem para atender todos os picos ao mesmo tempo [Armbrust et al., 2010]. Na nuvem colaborativa também ocorre a multiplexação estatística, mas isso evita que os próprios usuários precisem dimensionar seus recursos para o pico. Contudo, diferente das nuvens tradicionais, o caráter gratuito facilita que os recursos da nuvem colaborativa se esgotem. É essencial a presença de um mecanismo que decida as alocações quando a demanda por recursos supera a oferta. O mecanismo proposto neste trabalho evita o uso desmedido de recursos ao considerar os comportamentos passados no cálculo da credibilidade de cada usuário. Usuários que requisitam mais recursos do que fornecem são gradualmente penalizados nas decisões de alocação. Por outro lado, usuários com maiores contribuições no passado têm vantagens no conflito por recursos. O mecanismo funciona de acordo com os seguintes requisitos:

- **Racionalidade Individual**: As decisões de alocação devem ser cuidadosas de modo que nenhum usuário se prejudique em usar a nuvem colaborativa. Caso um usuário tivesse acesso a menos recursos do que teria sem fazer parte da nuvem, a decisão mais racional seria abandonar o sistema.
- **Eficiência Estrita de Pareto**: Os recursos do sistema devem ser usados de forma eficiente. O mecanismo deve garantir que o máximo de recursos do sistema seja utilizado.
- **Verdade**: Os usuários da nuvem não devem conseguir manipular o mecanismo de alocação ao mentir sobre suas necessidades por recursos.
- **Incentivo à Colaboração**: Os usuários devem obter benefícios por compartilhar seus recursos ociosos.
- **Competitividade de Novos Entrantes**: Usuários novos devem ter condição de competir por recursos do sistema, caso contrário, eles não se juntariam à nuvem.

O mecanismo proposto funciona em iterações. A cada iteração, os usuários expressam seu desejo de contribuir ou requisitar recursos. De acordo com os desejos e credibilidades de todos os usuários, o mecanismo pode decidir uma alocação de recursos. A credibilidade de cada usuário representa sua tendência em contribuir ou requisitar recursos. Considere que um usuário com credibilidade "-2" costuma contribuir 2 recursos, e

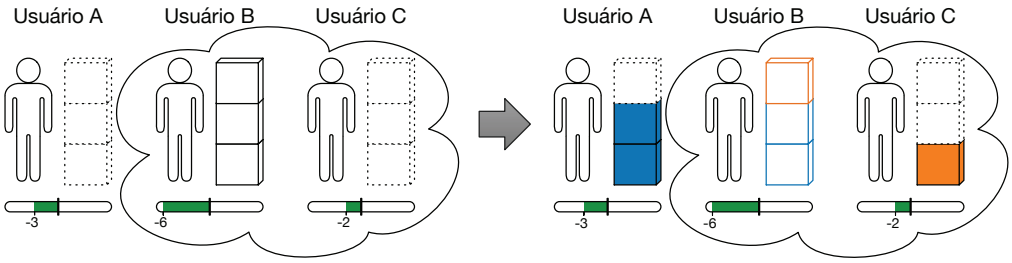


Figura 1. Exemplo da visão que o Usuário A tem da nuvem colaborativa. Os Usuários A e C precisam de 3 recursos cada, enquanto o Usuário B oferece 3 recursos. Como a credibilidade de A é -3 (barra verde inferior) e a de C é -2, o Usuário A recebe 2 recursos enquanto C recebe 1 recurso (1 a menos que A).

um usuário com credibilidade "-3" costuma contribuir 3 recursos. É natural assumir que, quando há sobrecarga da nuvem, o usuário que costuma contribuir com 3 recursos deve receber preferencialmente 1 recurso a mais em relação ao que costuma contribuir com 2 recursos. Essa intuição é a base do algoritmo de alocação proposto e está exemplificada na Figura 1.

3. Modelo de Compartilhamento de Recursos

No modelo, a utilização do sistema é dividida em intervalos de tempo de mesma duração. Em cada intervalo ocorre uma iteração em que os usuários escolhem a quantidade de recursos que desejam fornecer ou requisitar do sistema. O conjunto de usuários é denotado por $\mathcal{N}$, tal que $|\mathcal{N}| = N$. As escolhas dos usuários são representadas pelas chamadas ações. Uma ação de um usuário $i \in \mathcal{N}$ é denotada por a_i. Ações positivas representam pedidos por recursos enquanto que ações negativas representam contribuições. Como é impossível que o sistema saiba se a ação do usuário representa de fato o seu desejo (um usuário pode precisar de 2 recursos mas pedir 4 para o sistema), cada usuário possui um *tipo* que representa seu real desejo em contribuir ou requisitar recursos. O tipo de um usuário i é denotado por θ_i e só é conhecido por este usuário. Observe que se o usuário i revela a verdade, sua ação é igual ao seu tipo ($a_i = \theta_i$).

O modelo também assume a presença de um mecanismo. Com base nas ações dos usuários, o mecanismo deve ser capaz de decidir uma alocação de recursos. Um usuário i, recebe uma alocação de recursos o_i. Se o mecanismo decidiu que o usuário receberá recursos do sistema, sua alocação é positiva. Por outro lado, se o usuário contribuirá recursos para o sistema, sua alocação é negativa. Dependendo da alocação do mecanismo os usuários podem ficar satisfeitos ou não. Para cada usuário é definida também uma função utilidade. Denota-se u_i a função utilidade do usuário i. Quando a função utilidade retorna um valor positivo, o usuário está mais satisfeito do que estaria sem fazer parte da nuvem. Alternativamente, caso a função retorne um valor negativo, o usuário estaria mais satisfeito sem usar a nuvem. A seguir, as variáveis do modelo são definidas formalmente. O resumo das notações pode ser visto na Tabela 1.

3.1. Formalização do Modelo

O modelo é definido como um jogo Bayesiano, sendo dividido em cenário e mecanismo. Tudo que é definido no cenário é fixo e não pode ser redefinido pelo mecanismo. O cenário é modelado como uma tupla $(\mathcal{N}, \mathcal{O}, \Theta, p, u)$. $\mathcal{N}$ foi definido anteriormente e

Tabela 1. Resumo das notações usadas no modelo.

Notação	Descrição
N	Número de usuários
$\mathcal{N}$	Conjunto de todos os usuários
$\mathcal{O}$	Conjunto de alocações possíveis
o	Uma alocação de recursos em $\mathcal{O}$ para todos os usuários, $o = (o_1, \ldots, o_N)$
o_i	Quantidade de recursos alocados para i
Θ	Combinação de tipos possíveis de todos os usuários, $\Theta = \Theta_1 \times \ldots \times \Theta_N$
Θ_i	Conjunto de tipos possíveis para o usuário i
θ	Escolha de tipos de todos os usuários, $\theta \in \Theta$
θ_i	Um tipo de i, $\theta_i \in \Theta_i$
r_i	Quantidade de recursos que i possui
p	Distribuição de probabilidade dos tipos em Θ
u	Funções utilidade dos usuários, $u = (u_1, \ldots, u_N)$
$u_i(o, \theta_i)$	Função utilidade do usuário i
$\mathcal{A}$	Combinação de ações de todos os usuários, $\mathcal{A} = A_1 \times \ldots \times A_N$
A_i	Conjunto de ações possíveis para o usuário i
a	Ações escolhidas por todos os usuários, $a \in \mathcal{A}$
a_i	Ação escolhida por i
$\mathcal{M}(a)$	Função Mecanismo

representa o conjunto de usuários. $\mathcal{O}$ é o conjunto de alocações possíveis de forma que $(o_1, \ldots, o_N) \in \mathcal{O}$. As alocações de recursos devem garantir que a quantidade de recursos recebidos pelos usuários não ultrapasse a quantidade de recursos fornecidos. Além disso, é impossível que um usuário i forneça mais recursos do que de fato possui. Considerando que r_i representa a quantidade de recursos que o usuário i possui, o conjunto $\mathcal{O}$ pode ser definido como:

$$\mathcal{O} = \left\{ (o_1, \ldots, o_N) \in \mathbb{Z}^N \middle| (\forall i \in \mathcal{N}, o_i \geq -r_i) \wedge \sum_{i \in \mathcal{N}} o_i \leq 0 \right\}. \quad (1)$$

Θ é definida como $\Theta = \bigtimes_{i=1}^{N} \Theta_i$ em que Θ_i são os tipos possíveis para o usuário i. Os tipos dos usuários são limitados inferiormente pela quantidade de recursos que estes possuem de forma que $\Theta_i = \{\theta_i \in \mathbb{Z} | \theta_i \geq -r_i\}$. p representa a distribuição de probabilidade dos tipos em Θ. Finalmente, u representa uma tupla com as funções utilidade de todos os usuários, $u = (u_1, \ldots, u_N)$. A função utilidade do usuário i é definida como:

$$u_i(o_i, \theta_i) = \begin{cases} \min(o_i, \theta_i), \text{ para } \theta_i \geq 0. \\ \min(0, o_i - \theta_i), \text{ para } \theta_i < 0. \end{cases} \quad (2)$$

Observe na Equação 2 que quando um usuário precisa de mais recursos ($\theta_i > 0$) ele tem utilidade negativa caso $o_i < 0$ e utilidade positiva caso $o_i > 0$. No entanto a utilidade é limitada superiormente por θ_i, de modo que o usuário não se beneficia de mais recursos do que precisa. Alternativamente, quando um usuário possui recursos ociosos ($\theta_i < 0$) ele não pode obter utilidade positiva, mas obtem utilidade negativa caso $o_i < \theta_i$, ou seja, caso contribua com mais recursos do que gostaria. No caso intermediário em que $\theta_i = 0$, os usuários são indiferentes em receber recursos mas têm utilidade negativa caso contribuam com recursos para o sistema.

O mecanismo é um par $(\mathcal{A}, \mathcal{M})$. O conjunto $\mathcal{A}$ representa as possíveis escolhas de ações dos usuários do mecanismo. Como dito anteriormente, as ações dos usuários podem ou não ser iguais aos seus tipos. Ainda assim, as ações possíveis devem ser as mesmas que os tipos possíveis de forma que $\mathcal{A} = \Theta$.[1] $\mathcal{M}$ é a função mecanismo que mapeia as ações dos usuários em alocações de recursos ($\mathcal{M} : \mathcal{A} \rightarrow \mathcal{O}$). A função mecanismo independe da definição do modelo e será definida a seguir.

4. Mecanismo de Alocação

O mecanismo decide uma alocação de recursos válida a partir dos desejos dos usuários de receber ou contribuir. Para que haja um incentivo à contribuição de recursos ociosos, o mecanismo deve garantir que essa alocação privilegie usuários com maior contribuição no passado. As contribuições passadas são representadas pela credibilidade de cada usuário. No entanto, deve-se projetar o mecanismo de forma que também possibilite a entrada de novos usuários.

4.1. Credibilidade dos Usuários

A credibilidade do usuário i no instante t, representada por $c_i(t)$, é calculada usando uma média móvel exponencial das alocações passadas, definida de forma recursiva como: $c_i(t) = (1 - \delta) \cdot o_i(t) + \delta \cdot c_i(t - 1)$. A credibilidade usa o parâmetro $\delta \in [0, 1)$, o qual define o quanto as amostras passadas influenciam na credibilidade. Para $\delta = 0$, a credibilidade só considera a última alocação de recursos ($o_i(t)$). Quanto mais δ se aproxima de 1, mais os valores de alocações passadas influenciam na credibilidade. O amortecimento da credibilidade garante que novos usuários da nuvem, os quais entram no sistema com credibilidade nula, tenham condição de disputar recursos com os usuários mais antigos. Existe, contudo, um compromisso na escolha do valor de δ. Valores muito pequenos fazem com que o sistema não recompense suficientemente os usuários por suas alocações passadas. Por outro lado, valores muito próximos de 1 fazem com que a resposta do sistema às mudanças de comportamento dos usuários seja lenta.

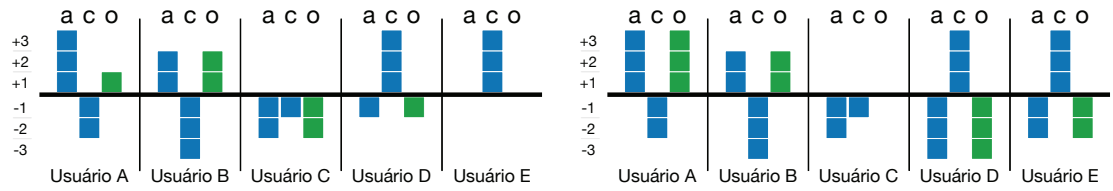
4.2. Restrição à Saída do Mecanismo

Embora o modelo limite as possíveis saídas do mecanismo ao conjunto $\mathcal{O}$, esse conjunto ainda admite a presença de algumas saídas indesejáveis de acordo com os requisitos do mecanismo. É conveniente definir um subconjunto $\Psi \subseteq \mathcal{O}$ com somente as alocações desejáveis:

$$\Psi = \left\{ (o_1, \ldots, o_N) \in \mathcal{O} \middle| \sum_{i \in \mathcal{N}} o_i = 0 \wedge \forall i \in \mathcal{N}, \min(a_i, 0) \leq o_i \leq \max(a_i, 0) \right\}. \quad (3)$$

A primeira restrição é contra o desperdício de recursos. Não é permitido que mais recursos sejam contribuídos do que utilizados. Ou seja, a alocação deve obedecer $\sum_{i \in \mathcal{N}} o_i = 0$. Outra restrição que se impõe tem a finalidade de garantir a racionalidade individual. Para isso, basta impedir alocações que gerem utilidade negativa, para qualquer usuário. Isso é feito ao garantir que $o_i \geq \min(\theta_i, 0), \forall i$. Para $a_i = \theta_i$, isso equivale a dizer

[1] Mecanismos onde $\mathcal{A} = \Theta$ são chamados em teoria dos jogos de mecanismos diretos. O princípio da revelação garante que não há perda de generalidade ao se impor essa restrição [Myerson, 1982].



(a) Demanda maior que oferta.

(b) Oferta maior que demanda.

Figura 2. Alocações (o) geradas pelo mecanismo a partir das ações (a) e credibilidades (c) dos usuários A até E. Ações positivas representam demanda, enquanto negativas representam oferta. Credibilidades negativas indicam que, no passado, o usuário contribuiu mais recursos do que requisitou.

que $o_i \geq \min(a_i, 0)$. Uma outra forma de desperdício de recursos é atribuir mais recursos a um usuário do que ele de fato precisa. Para impedir que isso ocorra, também com $a_i = \theta_i$, basta garantir que $o_i \leq \max(a_i, 0)$. A função mecanismo será então definida de forma a gerar saídas no conjunto Ψ.

4.3. Função Mecanismo

A intuição para a definição da função mecanismo está em inferir, através da credibilidade, a quantidade de recursos que um usuário costuma contribuir ou requisitar do sistema. Os usuários que costumam contribuir mais devem receber mais. Por outro lado, usuários que costumam contribuir menos poderão compensar futuramente com maior contribuição.

Ao garantir vantagens para os usuários que contribuíram mais no passado, surge um outro tipo de conflito: quando a oferta supera a demanda, quais usuários devem ser escolhidos para contribuir para o sistema e com isso ter uma melhora na credibilidade? Os dois tipos de conflito (oferta maior que demanda e demanda maior que oferta) são tratados pelo mecanismo de maneira simétrica. No caso do conflito pelo uso de recursos, o mecanismo privilegia os usuários com melhor credibilidade. No conflito pela oferta de recursos, o mecanismo privilegia os usuários com pior credibilidade. Assim, usuários que contribuíram mais no passado são preferencialmente recompensados com recursos. Já os que contribuíram com menos recursos têm maior chance de contribuir, podendo então melhorar suas credibilidades.

A Figura 2 ilustra um exemplo de dois cenários de conflito com cinco usuários de A até E. Para cada usuário, especifica-se os valores da ação (**a**), credibilidade (**c**) e alocação (**o**). No cenário onde a demanda por recursos é maior que a oferta (Figura 2(a)), existe demanda por 5 recursos, mas somente 3 recursos são oferecidos. O mecanismo cuida do conflito entre os usuários que precisam de recursos. Como o usuário B tem credibilidade melhor (mais negativa), ele contribuiu mais no passado e, portanto, recebe uma alocação de recursos maior. O mesmo exemplo pode ser adaptado de forma que a oferta supere a demanda (Figura 2(b)). Observe que agora todas as demandas são aceitas e o usuário C deixará de contribuir com recursos, já que sua credibilidade é melhor que a de D e de E, os quais conseguem suprir a demanda sozinhos.

Essas alocações podem ser computadas matematicamente como será mostrado a seguir. Considerando o primeiro exemplo (Figura 2(a)), há um vetor de ações $\boldsymbol{a} = [3, 2, -2, -1, 0]$ e um vetor de credibilidades, calculado a partir de alocações passadas

hipotéticas, $\boldsymbol{c} = [-2, -3, -1, 3, 3]$. O mecanismo usa $\boldsymbol{a}$ para definir o conjunto de alocações possíveis Ψ, conforme a Equação 3. Uma alocação $\boldsymbol{o} \in \Psi$ é então calculada de forma a minimizar a variância do vetor $\boldsymbol{c} + \boldsymbol{o}$ desde que garantindo a alocação da quantidade máxima de recursos. Em resumo, a função mecanismo resolve:

$$\underset{\boldsymbol{o} \in \Phi \subseteq \Psi}{\text{minimizar}} \sum_{i \in \mathcal{N}} (c_i + o_i)^2 \ \textit{(minimizar a variância)}, \tag{4}$$

tal que:

$$\Phi = \underset{\boldsymbol{o} \in \Psi}{\arg\max} \sum_{i \in \mathcal{N}} |o_i| \ \textit{(alocar a quantidade máxima de recursos)}. \tag{5}$$

O cálculo da variância não inclui a média do vetor $\boldsymbol{c} + \boldsymbol{o}$, já que a média é sempre nula[2]. Vale notar que minimizar a variância de $\boldsymbol{c} + \boldsymbol{o}$ neste caso equivale a minimizar os máximos de $\boldsymbol{c} + \boldsymbol{o}$ lexicograficamente (minimizar o primeiro máximo, o segundo e assim sucessivamente), quando a oferta é maior que a demanda, e maximizar os mínimos lexicograficamente, caso contrário. Para encontrar uma solução para a Equação 4 pode-se recorrer a métodos de otimização com restrições. No entanto, também é possível definir um algoritmo determinístico para resolver o mesmo problema.

4.4. Algoritmo de Alocação

As restrições criadas para a saída do mecanismo (Equações 3 e 5) fazem com que, quando a demanda é maior que a oferta, todas as ações de oferta são satisfeitas (se $a_i \leq 0$ então $o_i = a_i$). Já quando a oferta é maior que a demanda, todas as ações de demanda são satisfeitas (se $a_i \geq 0$ então $o_i = a_i$). Além de conveniente para o algoritmo, essa característica reforça a simetria entre os dois casos, sendo possível focar em apenas um deles. Quando a demanda é maior que a oferta $\left(\sum_{i \in \mathcal{N}} a_i > 0\right)$, o algoritmo precisa definir as alocações da parte em conflito ($a_i > 0$). A quantidade de recursos disponíveis a ser alocada para a parte em conflito é dada pela soma das ações negativas $\left(-\sum_{i \in \mathcal{N}} \min(a_i, 0)\right)$.

O Algoritmo 1 usa a ideia de maximizar os mínimos lexicograficamente. Para isso, faz-se uso de uma fila de prioridade Q, responsável por guardar as credibilidades dos usuários em conflito. Como as menores credibilidades são as melhores, a cada iteração calcula-se a diferença d entre a menor e a segunda menor credibilidades (linha 11). Para o usuário de menor credibilidade calcula-se uma alocação parcial como sendo o menor valor dentre a quantidade de recursos disponíveis r, o número de recursos requisitados pelo usuário $\boldsymbol{a}[i]$, e a diferença calculada d (linha 13). Essa forma evita não só que mais recursos sejam alocados do que disponíveis, mas também que o usuário não receba mais recursos do que precisa. Se a alocação parcial ainda não satisfizer as necessidades do usuário, sua credibilidade descontada pela alocação parcial é reinserida na fila (linha 19). Para o caso em que a oferta é maior que a demanda, é possível usar o mesmo algoritmo. Bastando inverter o sinal das entradas ($\boldsymbol{a} = [-a_1, \ldots, -a_N], \boldsymbol{c} = [-c_1(t), \ldots, -c_N(t)]$) e da saída ($\boldsymbol{o} = [-o_1, \ldots, -o_N]$). Para o caso excepcional em que que a demanda é igual à oferta, o algoritmo não é necessário, bastando fazer $\boldsymbol{o} = \boldsymbol{a}$.

[2]Por definição o somatório das alocações em Ψ é nulo e por consequência as credibilidades também.

Algoritmo 1 Alocação de recursos entre os usuários (demanda maior que a oferta).

Precondições: $\boldsymbol{a} \in \mathcal{A}$, $\boldsymbol{c} = [c_1(t), \ldots, c_N(t)]$
1: **para** $i \leftarrow 1, N$ **faça**
2: **se** $\boldsymbol{a}[i] > 0$ **então**
3: Inserir$(Q, \langle \boldsymbol{c}[i], i \rangle)$ ▷ Insere na fila Q as credibilidades dos usuários que precisam de recursos.
4: $\boldsymbol{o}[i] \leftarrow 0$
5: **senão**
6: $\boldsymbol{o}[i] \leftarrow \boldsymbol{a}[i]$ ▷ Todos os recursos oferecidos são alocados.
7: $r \leftarrow -\sum_{i=1}^{N} \min(\boldsymbol{a}[i], 0)$
8: **enquanto** $r > 0$ **faça** ▷ Enquanto há recursos disponíveis.
9: $\langle u, i \rangle \leftarrow \text{Extrair}_{\text{Min}}(Q)$ ▷ u é a melhor credibilidade.
10: $\langle v, j \rangle \leftarrow \text{Encontrar}_{\text{Min}}(Q)$ ▷ v é a segunda melhor credibilidade.
11: $d \leftarrow \lceil v - u \rceil$
12: **se** $d = 0$ **então** d $\leftarrow 1$
13: $\alpha \leftarrow \min(r, \boldsymbol{a}[i], d)$ ▷ Calcula uma alocação de recursos parcial.
14: $\boldsymbol{o}[i] \leftarrow \boldsymbol{o}[i] + \alpha$
15: $r \leftarrow r - \alpha$
16: **se** $\alpha < \boldsymbol{a}[i]$ **então** ▷ Usuário ainda precisa de mais recursos.
17: $u \leftarrow u + \alpha$
18: $\boldsymbol{a}[i] \leftarrow \boldsymbol{a}[i] - \alpha$
19: Inserir$(Q, \langle u, i \rangle)$ ▷ Reinsere o usuário na fila.
20: **retorne** $\boldsymbol{o}$

5. Avaliação Teórica

Esta seção mostra como o mecanismo satisfaz os requisitos introduzidos na Seção 2. Contudo, devido a restrições de espaço, as demonstrações foram deixadas para o relatório técnico [Sadok et al., 2016].

O requisito de Racionalidade Individual faz com que nenhum usuário obtenha vantagens ao deixar o mecanismo. Da forma como a função utilidade foi definida, isso equivale a garantir que nenhum usuário tenha utilidade negativa.

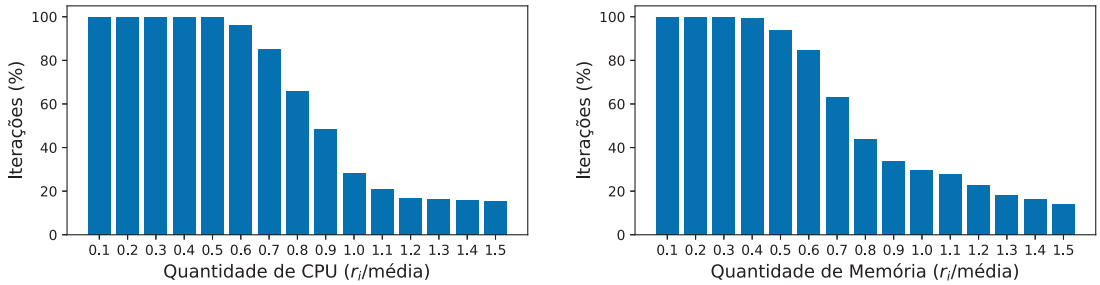
Teorema 1. ***Racionalidade Individual.*** *O mecanismo $(\mathcal{A}, \mathcal{M})$, em que $\mathcal{M}$ produz uma saída no conjunto Ψ (Equação 3), satisfaz a Racionalidade Individual. Ou seja, para todo i e $\boldsymbol{\theta}$, $u_i(\mathcal{M}(\boldsymbol{\theta}), \theta_i) \geq 0$.*

Para que o mecanismo satisfaça a Eficiência Estrita de Pareto é necessário que a alocação de recursos gere a maior soma de utilidades possível e que os usuários anunciem a verdade.

Teorema 2. ***Eficiência Estrita de Pareto.*** *Se o mecanismo $(\mathcal{A}, \mathcal{M})$ for verdadeiro e $\mathcal{M}$ satisfizer a restrição imposta pela Equação 5, então o mecanismo satisfaz a Eficiência Estrita de Pareto. O que significa dizer que, em equilíbrio, ele seleciona resultados $\boldsymbol{o} \in \mathcal{O}$ os quais satisfazem: $\sum_{i \in \mathcal{N}} u_i(\boldsymbol{o}, \theta_i) \geq \sum_{i \in \mathcal{N}} u_i(\boldsymbol{o}', \theta_i), \forall \boldsymbol{o}' \in \mathcal{O}$.*

O fato de o mecanismo ser verdadeiro, além de ser útil para garantir a Eficiência Estrita de Pareto, faz com que nenhum usuário consiga manipular o mecanismo. Os usuários são incentivados a falar a verdade caso eles não consigam ganhos de utilidade ao mentir.

Teorema 3. ***Verdade.*** *O mecanismo $(\mathcal{A}, \mathcal{M})$ em que $\mathcal{M}$ satisfaz a Equação 4 é verdadeiro. Ou seja, para todo i e θ_i, a estratégia no equilíbrio do usuário i é $a_i = \theta_i$.*



(a) Iterações com uso total de CPU.

(b) Iterações com uso total de memória.

Figura 3. Percentual de iterações onde o sistema se encontra sobrecarregado. Observe que, para uma parcela de até 0,5 da média de CPU ou 0,4 da média de memória, o sistema está completamente sobrecarregado.

Os demais requisitos não são definidos formalmente. Há um incentivo à colaboração uma vez que usuários com maior credibilidade recebem prioridade de alocação de recursos. Além disso, o fato da média das credibilidades dos usuários ser sempre zero faz com que os novos usuários entrem no sistema com a credibilidade igual à média, o que garante competitividade aos novos entrantes.

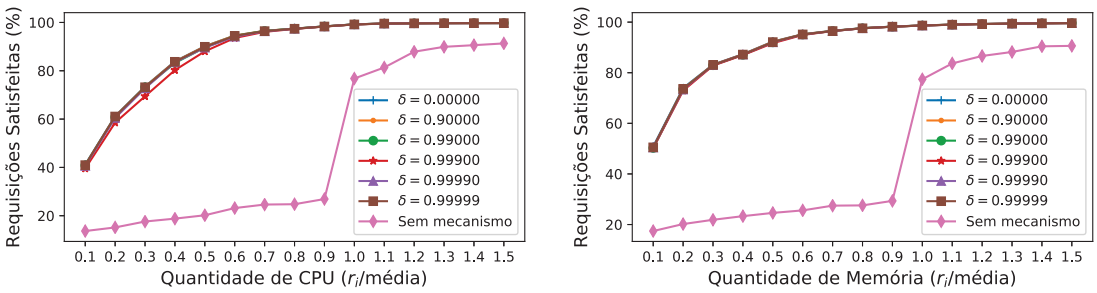
6. Avaliação Prática

Além da análise teórica, é feita uma análise com dados de uso de uma nuvem real. O conjunto de dados utilizado representa 29 dias na nuvem do Google, com aproximadamente 12.500 máquinas e 930 usuários [Reiss et al., 2011]. Os dados contêm informação das tarefas disparadas pelos usuários. Cada tarefa possui informação de início, término, identificador anônimo do usuário, além das quantidades requeridas de CPU e memória. Para as simulações é escolhido um conjunto de 100 usuários aleatórios e é usado um intervalo de 10 min entre cada iteração.[3] Além disso, uma vez que o Google normaliza a quantidade de CPU e memória requisitados por cada tarefa, esses valores são desnormalizados de forma que o menor valor presente no conjunto de dados equivalha a uma unidade do recurso. Eventuais valores racionais são arredondados para o menor inteiro maior ou igual (função teto).[4]

A partir das tarefas de cada usuário é possível inferir a quantidade de recursos que este precisa. Quando uma tarefa começa, incrementa-se a necessidade por recursos do usuário de acordo com a quantidade de recursos requeridos pela tarefa. Quando uma tarefa termina decrementa-se essa mesma quantidade. Contudo, há uma diferença importante em relação à posse de recursos: no Google, as máquinas são de propriedade da empresa, a qual possui autoridade sobre a alocação; enquanto nas nuvens colaborativas, cada usuário detém uma quantidade de recursos, hipoteticamente insuficiente para seus picos de uso. Como forma de aplicar o mesmo comportamento dos usuários ao cenário colaborativo, atribui-se a cada usuário uma quantidade de recursos. Cada usuário recebe uma quantidade de recursos proporcional à média de suas necessidades durante o período completo do conjunto de dados. Como forma de avaliar a proposta para diferentes

[3] 10 min é a duração mínima de uma tarefa na nuvem do Google.

[4] O código das simulações pode ser obtido em `https://www.gta.ufrj.br/~sadok/codes/credibility_allocation/`.



(a) Necessidades de CPU satisfeitas.

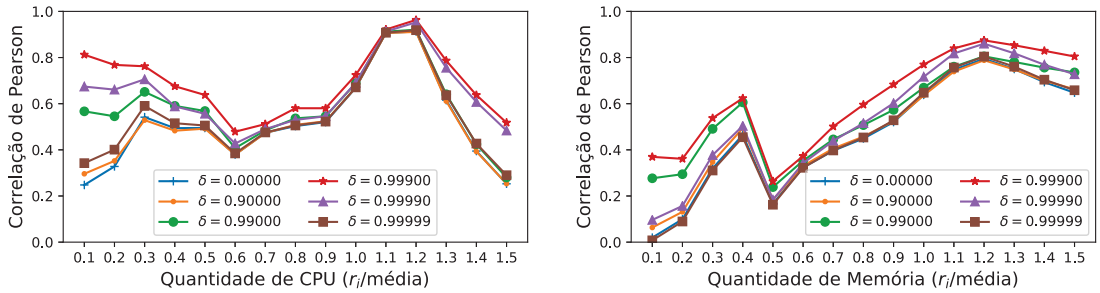
(b) Necessidades de memória satisfeitas.

Figura 4. Comparação das requisições atendidas ao usar ou não o mecanismo. O número de requisições atendidas é consideravelmente maior ao usar o mecanismo, independentemente do δ utilizado.

níveis de sobrecarga no sistema, esse fator de proporcionalidade é variado de 0,1 a 1,5 nos experimentos realizados. Quando o fator de proporcionalidade é igual a 1,0 os usuários recebem uma quantidade de recursos exatamente igual à média de suas necessidades. Uma vez definida a quantidade de recursos que cada usuário possui, os tipos dos usuários podem ser definidos. O tipo de um usuário em um determinado instante de tempo é a diferença entre a quantidade de recursos que este usuário possui e a sua necessidade por recursos nesse mesmo instante. Quando a necessidade do usuário supera os seus recursos, o tipo se torna positivo e ele pede recursos. Quando a necessidade é menor que a quantidade de recursos do usuário, o tipo se torna negativo e o usuário oferece recursos.

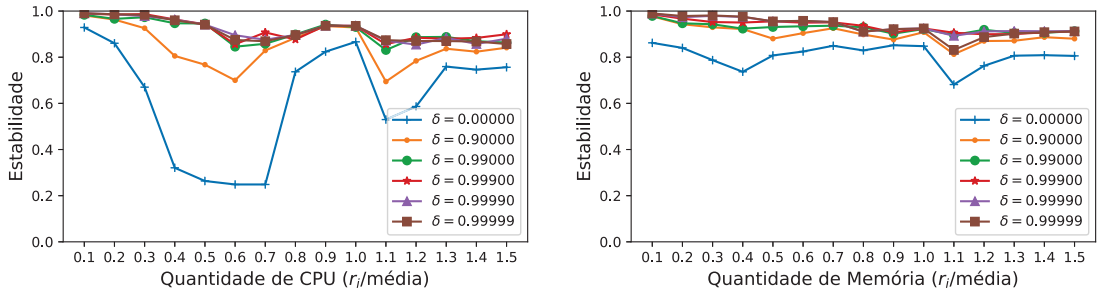
6.1. Desempenho Global

Primeiro, analisa-se o sistema para verificar o desempenho global, ou seja, qual a eficiência no uso de recursos e qual o impacto na satisfação geral dos usuários. A Figura 3 mostra a sobrecarga do sistema para quantidades diferentes de recursos atribuídos a cada usuário. Quando os usuários possuem uma quantidade de recursos até 0,5 da média de CPU, todos os recursos de CPU são usados em todas as iterações. O mesmo ocorre até 0,4 da média de memória. Já para mais de 0,5 da média de CPU e memória o sistema não se encontra mais sobrecarregado o tempo todo. Naturalmente, conforme os usuários possuem mais recursos próprios, a sobrecarga no sistema diminui; com a diminuição da sobrecarga, mais requisições por recursos são atendidas pelo sistema. A Figura 4 mostra o percentual de requisições atendidas para diferentes valores do parâmetro δ e para diferentes níveis de sobrecarga do sistema. Além disso compara-se com o percentual de requisições atendidas caso os usuários não fizessem parte do mecanismo, usando apenas os próprios recursos. O número de requisições atendidas ao usar o mecanismo é consideravelmente maior e é pouco influenciada pelo parâmetro δ. Nota-se também que, sem o mecanismo, há um aumento brusco no número de requisições satisfeitas para 1,0 da média de recursos atribuídos, tanto para CPU quanto para memória. Isso decorre do fato de muitas requisições estarem próximas da média. Assim, quando os usuários possuem mais recursos do que a média de suas necessidades, essas requisições passam a ser atendidas. No entanto, mesmo com o salto de requisições atendidas, as requisições satisfeitas sem usar o mecanismo demoram a se aproximar de 100% devido a incidência frequente de picos de demanda, os quais não podem ser atendidos sem o uso da nuvem.



(a) Correlação entre CPU contribuída e recebida. (b) Correlação entre memória contribuída e recebida.

Figura 5. Correlação de Pearson entre os recursos contribuídos e recebidos no mecanismo. Observe como $\delta = 0{,}999$ resulta na melhor correlação para todas as parcelas de recursos atribuídos.



(a) Estabilidade da alocação de CPU. (b) Estabilidade da alocação de memória.

Figura 6. Mediana das Autocorrelações *lag-1* das alocações de CPU e memória de todos os usuários.

6.2. Influência do Parâmetro δ

O valor de δ determina o quanto as alocações passadas influenciam as credibilidade dos usuários. Como foi visto, ele não influencia o número de requisições atendidas no sistema. Contudo, espera-se que valores de δ próximos de 0 façam com que os usuários não tenham suas contribuições passadas devidamente recompensadas, enquanto valores de δ muito próximos de 1 devem fazer com que a credibilidade demore a convergir e não represente as contribuições dos usuários. Os valores de δ são variados em intervalos logarítmicos já que a média móvel exponencial também é influenciada logaritmicamente. Uma forma de observar a influência do valor de δ na recompensa dos usuários é calcular o coeficiente de correlação de Pearson entre os recursos contribuídos e recebidos. Quanto mais próximo de 1 for o coeficiente, maior a correlação, o que indica a tendência dos usuários que contribuíram mais recursos também receberem mais. A Figura 5 mostra a correlação de Pearson para os diferentes valores de δ e para diferentes níveis de sobrecarga do sistema. A maior correlação, em todos os níveis de sobrecarga e para os dois tipos de recursos foi para $\delta = 0{,}999$. Como esperado, valores de δ muito próximos de 1 (como $\delta = 0{,}99999$) ou muito pequenos (como $\delta = 0$) apresentaram desempenho inferior.

O valor de δ também influencia a estabilidade das alocações. Valores de δ mais próximos de 0 fazem com que as credibilidades se alterem mais facilmente, fazendo com que as decisões de alocação do mecanismo também mudem mais facilmente. Essas osci-

lações degradam o desempenho do sistema. Para medir a influência de δ na estabilidade foi usada a autocorrelação *lag-1* ($\mathcal{R}_1$) da alocação de recursos, calculada em função das alocações $o_i(t)$ do usuário i em cada instante t: $\mathcal{R}_1(i) = \frac{\sum_{t=1}^{n-1}(o_i(t)-\bar{o_i})(o_i(t+1)-\bar{o_i})}{\sum_{t=1}^{n}(o_i(t)-\bar{o_i})^2}$, onde n é o número de iterações na duração do conjunto de dados analisado e $\bar{o_i}$ é a média das alocações do usuário i. Quanto mais próximo de 1 for $\mathcal{R}_1$, mais estáveis são as alocações. A Figura 6 compara as medianas das autocorrelações *lag-1* das alocações de todos os usuários para diferentes valores de δ e de sobrecarga do sistema. Observe como, para os menores valores de δ (0,0 e 0,9), as alocações foram menos estáveis.

7. Trabalhos Relacionados

O compartilhamento de recursos em sistemas par-a-par foi notavelmente analisado por Buragohain et al. [Buragohain et al., 2003] usando teoria dos jogos. A proposta busca aumentar a reciprocidade de contribuição de recursos entre pares de nós, considerando a falta de confiança entre os usuários do sistema. Um outro trabalho com objetivo semelhante é o FD-NoF [Falcão et al., 2015], que usa uma rede de favores para promover justiça através da contenção de recursos dos usuários de uma nuvem par-a-par. Embora a contenção de recursos auxilie na promoção de justiça, isso gera perda da eficiência, agravada ainda mais por não se considerar as reciprocidades indiretas entre os usuários. Diferentemente das propostas para sistemas par-a-par, a centralização do controle nas nuvens colaborativas permite a presença de um sistema único de reputação, onde reciprocidades indiretas podem ser consideradas. O presente trabalho usa essa vantagem, promovendo a máxima eficiência do uso dos recursos e, ao mesmo tempo, incentivando a colaboração entre os usuários do sistema.

Os trabalhos que analisam a possibilidade de coalizão entre provedores de nuvem também possuem semelhanças com este trabalho. Samaan [Samaan, 2014] criou um modelo para definir preços e a quantidade de recursos que cada provedor de nuvem deve contribuir para um mercado de excedentes (*spot market*). Também é usado um modelo com jogo repetido, mas o equilíbrio foi garantido com uma estratégia de *grim trigger*. Niyato et al. [Niyato et al., 2011] também tratam de coalizões entre provedores de nuvem mas analisam as utilidades totais para diferentes combinações de provedores, o modelo da proposta exige o uso de pagamento entre as partes para formar as coalizões. Este trabalho não assume a presença de pagamento entre as partes e propõe um equilíbrio sem usar estratégias severas como o *grim trigger*. Além disso, diferente dos outros trabalhos, a proposta é validada também usando um traço real de utilização de uma nuvem.

8. Conclusões

As nuvens colaborativas permitem que usuários com poucos recursos possam usar recursos ociosos de outros usuários. Embora os ganhos de eficiência sejam claros, é necessário a presença de um mecanismo para reger as alocações e garantir um bom equilíbrio para o sistema. Foi proposto um mecanismo que rege os pedidos dos usuários com base em suas respectivas credibilidades. As alocações de recursos privilegiam os pedidos de recursos dos usuários com as melhores credibilidades e as requisições de doação de recursos dos usuários de pior credibilidade. Foi mostrado que o sistema faz uso eficiente dos recursos, incentiva a colaboração e a honestidade dos participantes. Além disso, o sistema de credibilidade com média nula dá oportunidade de novos usuários se integrarem à nuvem. A avaliação com dados reais de utilização permitiu verificar o desempenho

da proposta para diferentes valores do parâmetro do mecanismo. Como trabalho futuro, pretende-se usar o mesmo mecanismo para outras aplicações que envolvam o compartilhamento de recursos como, por exemplo, redes elétricas inteligentes.

Referências

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. e Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50.

Buragohain, C., Agrawal, D. e Suri, S. (2003). A game theoretic framework for incentives in P2P systems. *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*, 0121562:48–56.

Costa, L. H. M. K., de Amorim, M. D., Campista, M. E. M., Rubinstein, M. G., Florissi, P. e Duarte, O. C. M. B. (2012). Grandes massas de dados na nuvem: Desafios e técnicas para inovação. *Minicursos do Simpósio Brasileiro de Redes de Computadores*, p. 58.

Couto, R. S., Sciammarella, T., Barreto, H. F. S. S. M., Campista, M. E. M., Rubinstein, M. G., Costa, L. H. M. K., Vetter, F. e Marins, A. (2015). GT-PID: Uma nuvem IaaS universitária geograficamente distribuída. *TICAL2015*, p. 19.

Falcão, E. d. L., Brasileiro, F., Brito, A. e Vivas, J. L. (2015). Controlando a contenção de recursos para promover justiça em uma federação peer-to-peer de nuvens privadas. *XIII Workshop em Clouds e Aplicações (WCGA2015)*.

Myerson, R. B. (1982). Optimal coordination mechanisms in generalized principal–agent problems. *Journal of mathematical economics*, 10(1):67–81.

Niyato, D., Vasilakos, A. V. e Kun, Z. (2011). Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach. *Proceedings - 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, p. 215.

Reiss, C., Wilkes, J. e Hellerstein, J. L. (2011). Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA. Revisado em 2012-03-20. `http://code.google.com/p/googleclusterdata/wiki/TraceVersion2`.

Sadok, H., Campista, M. E. M. e Costa, L. H. M. K. (2016). Um mecanismo para compartilhamento de recursos em nuvens colaborativas baseado na credibilidade dos usuários. Relatório Técnico GTA-16-47, GTA/PEE/UFRJ.

Samaan, N. (2014). A Novel Economic Sharing Model in a Federation of Selfish Cloud Providers. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):12–21.

Sotomayor, B., Montero, R. S., Llorente, I. M. e Foster, I. (2009). Virtual Infrastructure Management in Private and Hybrid Clouds. *IEEE Internet Computing*, 13(5):14–22.

Vaquero, L. M., Rodero-Merino, L., Caceres, J. e Lindner, M. (2008). A break in the clouds. *ACM SIGCOMM Computer Communication Review*, 39(1):50.

Weins, K. (2016). Cloud computing trends: 2016 state of the cloud survey. `http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2016-state-cloud-survey`. Acessado: 2016-09-19.