Implementação e Avaliação de um Mecanismo de Escambo para Promover Reciprocidade em Federações de Provedores de Computação na Nuvem

Eduardo de Lucena Falcão, Francisco Brasileiro, Andrey Brito, Francisco Germano

[1]Universidade Federal de Campina Grande
Departmento de Sistemas e Computação
Campina Grande, Brasil

eduardolfalcao@lsd.ufcg.edu.br, {fubica,andrey}@computacao.ufcg.edu.br

chicog@lsd.ufcg.edu.br

Resumo. *Um grande desafio na concepção e operação de uma federação de provedores de computação na nuvem é a implementação de um modelo de negócio adequado que viabilize o funcionamento da federação. Modelos de negócio baseados em escambo foram propostos como uma forma barata e eficiente para mediar as transações entre os provedores que compõem a federação. Neste trabalho, discutimos os desafios relacionados com a implementação de um mecanismo de escambo em uma federação de provedores de computação na nuvem, proposto por nós anteriormente. O desempenho do mecanismo é avaliado através de experimentos controlados. Os resultados obtidos corroboram os resultados previamente obtidos com modelos de simulação simplificados. Em particular, o mecanismo implementado é capaz de garantir a paridade entre recursos consumidos e doados, ao mesmo tempo que mantém a satisfação dos membros colaboradores em níveis suficientemente altos.*

Abstract. *A big challenge when designing and operating a federation of cloud computing providers is the implementation of a suitable business model that enables the federation to function appropriately. Barter-based business models were proposed as a cheap and efficient way to mediate transactions between the providers that form the federation. In this work, we discuss the challenges related to the implementation of a barter mechanism in a federation of cloud computing providers, proposed by us previously. The performance of the mechanism is evaluated through controlled experiments. The results obtained corroborate previous results obtained with simplified simulation models. In particular, the mechanism in place is able to ensure fairness in the use of resources while maintaining the satisfaction of collaborative members at sufficiently high levels.*

1. Introdução

Participar de uma federação de nuvens pode ajudar provedores públicos ou privados a se organizarem melhor para elevar a qualidade de serviço ofertada a seus clientes. A principal ideia é que membros que estejam experimentando uma sobrecarga possam terceirizar a carga excedente para outros membros que estejam com baixa demanda. Isto possibilita

que os provedores de nuvem sirvam seus clientes adequadamente, mesmo em momentos de pico. Por outro lado, os provedores reduzem a ociosidade de suas infraestruturas ao servirem requisições de membros sobrecarregados, elevando, porém, seus custos operacionais. Dessa forma, tal terceirização precisa ser compensada de alguma forma.

No que concerne sua organização, é possível descrever o estilo de uma federação de nuvens baseado em três aspectos inter-relacionados: sua *arquitetura* [Grozev and Buyya 2014], o *modelo de negócio* [Petri et al. 2015] utilizado, e os *acordos de níveis de serviço/federação* (SLA ou FLA, do inglês *Service/Federation Level Agreement*) [Assis and Bittencourt 2016].

Dependendo da configuração desses três atributos, uma federação pode variar entre extremamente flexível a extremamente rígida, com regras bem definidas – características essencialmente opostas, cada uma com sua vantagem: liberdade para a primeira, e previsibilidade (ou menor risco) para a segunda. Por exemplo, é comum que arquiteturas centralizadas e modelos de negócios baseados na existência de uma moeda comum — abordagens monetárias — sejam tipicamente pouco flexíveis, já que toda comunicação e mediação comercial é realizada por uma entidade centralizada — uma espécie de banco — cuja customização é restrita ao administrador da federação. Com relação aos SLAs, os níveis de serviço da federação podem ser regidos através de acordos bilaterais entre as partes, assegurando a qualidade de serviço especificada, ou simplesmente inexistir, funcionando através de uma abordagem de melhor esforço.

O foco desse trabalho são federações altamente customizáveis (arquitetura descentralizada), desburocratizadas (sem SLAs), e de livre mercado (baseado em escambo). Arquiteturas descentralizadas geralmente dão mais liberdade de configuração para os membros da federação e, aliadas à ausência de um contrato de ingresso, facilitam a entrada de novos membros no momento que lhes aprouver. O livre mercado permite que os participantes permutem recursos baseados nas métricas que lhes convirem, não sofre interferência de entidade reguladora, e por funcionar de modo distribuído, têm baixo custo operacional e escalam com facilidade. Nós acreditamos que essa liberdade possa atrair mais participantes para a federação, melhorando sua eficiência, uma vez que aumenta a probabilidade de diferentes membros estarem em um momento de vale e pico de demanda, o que permite a cooperação.

Apesar de arquiteturas ponto-a-ponto (P2P, do inglês *Peer-to-Peer*) descentralizadas serem mais flexíveis, extensíveis e escaláveis, existem algumas dificuldades relacionadas à descoberta, roteamento, segurança e confiabilidade. Para o estilo de federação proposto neste trabalho, a confiabilidade está intimamente relacionada com o mecanismo de escambo. O livre mercado impõe que consumidores e provedores, que *a priori* são desconhecidos, explorem o mercado e negociem diretamente. Se nenhum incentivo à cooperação (ou punição por falta de cooperação) for concedido, é natural que alguns participantes prefiram agir como "caronas", *i.e.*, indivíduos que apenas consomem recursos, sem nada oferecer em troca, tornando a federação infrutífera.

O problema da alocação eficiente de recursos já foi extensivamente investigado e aplicado em sistemas P2P para compartilhamento de arquivos e grades computacionais P2P. A maioria das estratégias foca na priorização de participantes de acordo com métricas como reputação ou reciprocidade, e obviamente, isto traz incentivos à cooperação. De

fato, a maioria dessas abordagens asseguram aos participantes altos níveis de **satisfação** – a razão entre a quantidade de recursos consumidos e requisitados. Contudo, em cenários de baixa contenção de recursos, *i.e.*, cenários com abundância de recursos, a priorização por si só não é suficiente para marginalizar os caronas. Nesses cenários, os caronas conseguem consumir os recursos excedentes e nunca os retribuem, o que afeta a **paridade** dos membros cooperativos – a razão entre a quantidade de recursos consumidos e doados.

Em grades computacionais o custo de fornecimento dos recursos excedentes não é muito maior do que mantê-los ociosos e, portanto, a paridade não é mais importante do que a satisfação nestes cenários. Quando o contexto é uma federação P2P de provedores de computação na nuvem as prioridades passam a ser diferentes. As nuvens proveem recursos dedicados e em larga escala, o que torna a paridade um fator relevante.

O problema dos baixos níveis de paridade em cenários com baixa contenção de recursos já foi abordado em trabalhos anteriores [Falcão et al. 2015, Falcão et al. 2016a] através da reutilização da Rede de Favores (NoF, do inglês *Network of Favors*) [Andrade et al. 2007], um mecanismo de incentivo à cooperação baseado em reciprocidade direta, pensado originalmente para o contexto de grades computacionais oportunistas. Para melhor adequação ao contexto de federações de nuvens, onde a paridade é tão importante quanto a satisfação, a NoF foi estendida com um mecanismo de laço de controle retroalimentado, executado por cada participante, cujo trabalho é monitorar a paridade bilateral entre um membro P e os outros membros da federação, para decidir a quantidade de recursos que deve ser ofertada por P a cada outro membro da federação. O objetivo é evitar a provisão de recursos para membros não-recíprocos, e portanto, alcançar níveis adequados de paridade. Esta é a razão pela qual este esquema é chamado de Rede de Favores Dirigida à Paridade (FD-NoF, do inglês *Fairness-Driven NoF*).

Resultados de simulações da FD-NoF em cenários com baixa contenção de recursos sugerem que os participantes que executam o laço de controle retroalimentado conseguem elevar seus níveis de paridade ao mesmo tempo que mantém bons níveis de satisfação [Falcão et al. 2016a]. O objetivo deste trabalho é discutir os desafios envolvidos com a implementação da FD-NoF em um *middleware* para federação de provedores de computação na nuvem e avaliar o seu desempenho. Através dessa implementação e de sua experimentação em um ambiente controlado é possível evidenciar problemas mascarados pelo modelo de simulação simplificado.

O restante deste trabalho está organizado da seguinte forma. Os trabalhos relacionados são apresentados na próxima seção. A Seção 3 descreve brevemente o funcionamento da Rede de Favores Dirigida à Paridade, enquanto que a Seção 4 apresenta a sua implementação no *middleware* Fogbow [Brasileiro et al. 2016]. Na Seção 5 são explicados o projeto de experimentos, a infraestrutura utilizada e a metodologia de coleta dos resultados. Os resultados dos experimentos e suas correspondentes análises são apresentados na Seção 6. A Seção 7 encerra este trabalho apresentando as principais conclusões e direções para trabalhos futuros.

2. Trabalhos Relacionados

A grande maioria dos trabalhos sobre sistemas P2P para compartilhamento de arquivos usam mecanismos de reciprocidade baseados em escambo como forma de promover cooperação, visto que o recurso compartilhado – banda de rede – tipicamente não impõe

custos adicionais associados à taxa de uso do recurso. Por esta razão, a maioria dos mecanismos de reciprocidade usados em sistemas de compartilhamento de arquivos têm como objetivo principal assegurar altos níveis de satisfação aos participantes, não se importando com os níveis de paridade.

A ideia principal desses mecanismos é basear-se em alguma métrica de reciprocidade, de forma direta ou indireta, para decidir qual participante deve ser priorizado em um instante qualquer. O mecanismo proposto por Cohen (2003) é baseado em reciprocidade direta por permuta imediata, também conhecido como *tit-for-tat*, interação na qual os dois indivíduos devem possuir recursos de interesse em um mesmo momento. A vantagem da reciprocidade direta é que as informações sobre os níveis de reciprocidade dos demais participantes são coletadas e mantidas de forma individual, *i.e.*, são íntegras, e o imediatismo do mecanismo permite identificar e punir os caronas rapidamente. Ao contrário da estratégia proposta por Cohen (2003), a Rede de Favores [Andrade et al. 2007] utiliza a permuta tardia e prioriza os consumidores através de um sistema de créditos.

Com relação a mecanismos de incentivo à cooperação baseados em reciprocidade indireta, as principais estratégias utilizam sistemas de reputação [Kamvar et al. 2003] e sistemas de crédito transitivo [Menasché et al. 2010, Falcão et al. 2016b]. Ao contrário da reciprocidade direta, na reciprocidade indireta, as informações acerca dos comportamentos dos demais participantes são coletadas em terceiros, dando margem para conluio entre nós mal intencionados. Contudo, o *bootstrapping* dos participantes em comunidades baseadas em reciprocidade indireta é mais eficiente, pois a base de informações cresce rapidamente, e isso facilita a interação em comunidades nas quais os participantes possuem recursos diversificados. Em suma, a reciprocidade indireta pode ser uma solução adequada para comunidades nas quais os participantes interagem esporadicamente.

No que concerne comunidades de compartilhamento de recursos computacionais, especificamente as grades computacionais, já foram utilizadas abordagens monetárias [Kumar et al. 2011], abordagens baseadas em reciprocidade direta através de sistemas de créditos [Andrade et al. 2007], e reciprocidade indireta por reputação [Zhao and Li 2009]. Em relação a federações de nuvens, um *survey* realizado por Assis e Bittencourt (2016) aponta que os trabalhos adotam abordagens monetárias [Gomes et al. 2012], ou estratégias baseadas em SLAs [Kecskemeti et al. 2012].

A adoção de abordagens monetárias decorre da preocupação em assegurar paridade quando os recursos compartilhados não têm custo desprezível. Porém, apesar de ser eficiente, uma vez que os caronas teriam que pagar pelos recursos, impõe certa burocracia e requer alguns procedimentos complexos para garantir segurança. Esta é a lacuna que a Rede de Favores Dirigida à Paridade (FD-NoF) [Falcão et al. 2015, Falcão et al. 2016a] visa preencher: um mecanismo de escambo baseado em reciprocidade direta que assegura paridade em cenários de baixa contenção de recursos, possui implementação simples, funcionamento leve, e não burocratiza a federação. Resultados iniciais baseados em um modelo de simulação simplificado indicam que a FD-NoF consegue garantir paridade, ao mesmo tempo que mantém a satisfação dos membros colaborativos suficientemente alta, quando existem recursos disponíveis para serem compartilhados. Neste trabalho nós focamos nos aspectos de implementação da FD-NoF em um contexto específico e na avaliação dessa implementação através da execução de experimentos controlados.

3. A Rede de Favores Dirigida à Paridade (FD-NoF)

Na Rede de Favores (NoF), um favor consiste na alocação de um recurso para um nó solicitante por um período de tempo, gerando um crédito para o nó provedor e um débito para o nó consumidor. A NoF usa um sistema de saldos para priorizar os nós mais recíprocos, explicado a seguir.

Assumindo que $\upsilon(A, B, t)$ representa o valor total dos favores que um nó A doou para outro nó B até um dado momento t, e que $\gamma(A, B, t)$ é uma função que denota o saldo de B na visão de A exatamente após o tempo t, a definição mais simples para a função de computo do saldo é $\gamma(A, B, t) = \upsilon(B, A, t) - \upsilon(A, B, t)$. Para evitar que nós maliciosos sejam capazes de manipular seu saldo alterando sua identidade e aparecendo como um nó recém-chegado na comunidade, a NoF impede que os saldos assumam valores negativos através da seguinte equação: $\gamma(A, B, t) = max\left\{0, \upsilon(B, A, t) - \upsilon(A, B, t)\right\}$. Entretanto, utilizando esta equação para calcular o saldo, um nó A não conseguiria distinguir um nó B carona que nunca doou recursos, de um nó cooperativo C que já doou para A no passado mas consumiu de A pelo menos a mesma quantidade que doou. Por esta razão, Andrade *et al.* (2007) introduziram na definição da função de saldo $\gamma(A, B, t)$ uma parcela histórica que leva em consideração a quantidade amortizada de doações que A recebeu de B no passado: $\gamma(A, B, t) = max\left\{0, \upsilon(B, A, t) - \upsilon(A, B, t) + \log \upsilon(B, A, t)\right\}$.

Mecanismos de reciprocidade por permuta tardia se baseiam na expectativa de que os participantes irão eventualmente retribuir os favores recebidos. Portanto, na NoF, os nós sempre ofertam todos os seus recursos ociosos, na expectativa de acumular créditos com outros participantes e, por isto, neste trabalho, é chamada de Rede de Favores Dirigida à Satisfação (SD-NoF, do inglês *Satisfaction-Driven NoF*). Esta abordagem é extremamente eficiente em termos de satisfação e paridade sempre que a demanda dos nós cooperativos for maior ou igual à quantidade de recursos ofertados na federação. Porém, em cenários de baixa contenção de recursos, a paridade dos nós cooperativos é afetada. Para solucionar este problema, Falcão *et al.* (2016) sugeriram que a NoF fosse estendida com um mecanismo que regulasse a quantidade de recursos a ser ofertada de acordo com a reciprocidade dos demais participantes. Esta abordagem é chamada de Rede de Favores Dirigida à Paridade (FD-NoF), e seu funcionamento é descrito a seguir, após a formalização do conceito de paridade.

A paridade de um nó B na perspectiva de A é definida pela razão entre a quantidade total de recursos que A consumiu de B e a quantidade total de recursos que o mesmo nó A doou para B, até um dado momento. Para os casos em que A tenha realizado pelo menos uma doação para B até um dado momento t, essa função pode ser definida como $\phi(A, B, t) = \frac{\upsilon(B,A,t)}{\upsilon(A,B,t)}$. Por conveniência, assume-se que $\upsilon(A, A, t) = 1, \forall A \in \mathbb{F}$, onde $\mathbb{F}$ denota o conjunto de todos os participantes da federação. Quando medida em relação à federação como um todo, a paridade é definida pela função $\phi : \mathbb{F} \times \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ que, sempre que um nó A tenha provido algum recurso a algum outro nó, pode ser definida por $\phi(A, t) = \dfrac{\sum_{P \in \mathbb{F}} \upsilon(P, A, t)}{\sum_{P \in \mathbb{F}} \upsilon(A, P, t)}$.

Assuma que cada nó cooperativo tenha capacidade total de recursos $\mathcal{C} \in \mathbb{N}$, e que a quantidade de recursos que um nó A disponibiliza para qualquer outro nó B seja expressa pela função $\alpha : \mathbb{F} \times \mathbb{F} \times \mathbb{N} \rightarrow \mathbb{N}$. Nesse sentido, o objetivo do mecanismo é sugerir para A a quantidade de recursos $\alpha(A, P, t), \forall P \in \mathbb{F} - \{A\} \wedge \forall t \in \mathbb{N}$ que deve ser

ofertada para que A alcance os patamares desejados de paridade.

O controlador de capacidade proposto requer que cada nó cooperativo A defina dois valores, um limite mínimo $\mathcal{L}_{min} \in \mathbb{R}_{\geq 0}$ e um limite máximo $\mathcal{L}_{max} \in \mathbb{R}_{\geq 0}$, $\mathcal{L}_{max} \geq \mathcal{L}_{min}$, estabelecendo portanto um intervalo $[\mathcal{L}_{min}, \mathcal{L}_{max}]$ que denota os níveis desejados de paridade em relação aos outros membros da federação. O controlador funciona da seguinte maneira. Inicialmente, $\alpha(A, P, 0) = \mathcal{C}, \forall P \in \mathbb{F}$. Se em algum momento a paridade de A em relação a um nó B for inferior ao limite mínimo, *i.e.*, $\phi(A, B, t) < \mathcal{L}_{min}$, o controlador de capacidade irá continuamente subtrair uma constante Δ de $\alpha(A, B, t)$ (enquanto $\alpha(A, B, t) > 0$) nos passos de tempo subsequentes até que $\phi(A, B, t)$ seja superior a $\mathcal{L}_{min}$. Por outro lado, se $\phi(A, B, t) > \mathcal{L}_{max}$, o controlador de capacidade ofertada somará uma constante Δ ao valor de $\alpha(A, B, t)$ (enquanto $\alpha(A, B, t) < \mathcal{C}$) nos passos de tempo subsequentes até que $\phi(A, B, t)$ assuma um valor inferior a $\mathcal{L}_{max}$. Finalmente, quando $\mathcal{L}_{min} \leq \phi(A, B, t) \leq \mathcal{L}_{max}$, o controlador de capacidade executará um algoritmo de subida de encosta que consiste em utilizar os valores mais recentes de paridade para decidir se a capacidade ofertada deve ser aumentada ou diminuída, para maximizar a paridade de A com B dentro do intervalo $[\mathcal{L}_{min}, \mathcal{L}_{max}]$.

A abordagem descrita não se aplica a um nó B para o qual A nunca doou recursos, uma vez que neste caso $\phi(A, B, t)$ é indefinida. Nestes casos, A determina $\alpha(A, B, t)$ rodando o controlador de capacidades usando sua paridade em relação à federação como um todo. Note que isto só irá acontecer depois que A tenha feito sua primeira doação a algum nó; até este momento, A ofertará toda sua capacidade ociosa ($\mathcal{C}$) aos solicitantes.

4. Projeto e Implementação do Sistema

Como mencionado anteriormente, a FD-NoF foi implementada para ser usada por federações que utilizam o *middleware* Fogbow [Brasileiro et al. 2016]. Nesse tipo de federação, cada nuvem executa um agente Fogbow local (programa Java), que implementa a interface de acesso à federação e a comunicação entre os agentes. Cada agente Fogbow é composto por um componente principal e *plugins* comportamentais, que oferecem uma maneira simples de customização das principais operações realizadas pelo agente. Na prática, esses *plugins* são interfaces (Java) com funções bem definidas que podem ser implementados pelos administradores de cada nuvem como lhes convirem.

Para explicar como a FD-NoF foi implementada no Fogbow é preciso primeiro descrever o fluxo que uma requisição de alocação de recursos percorre, desde o momento de sua submissão, até o momento em que ela é atendida. Nessa descrição nós explicamos os *plugins* envolvidos e como eles foram customizados para implementar a FD-NoF.

A Figura 1 ilustra os dois principais processos relacionados ao escambo de recursos: o controle de admissão de requisições e a gerência de *orders*. Cada *order* carrega consigo seu estado e os requisitos básicos sobre memória e processamento. Sempre que uma requisição é admitida o agente cria uma *order* e a adiciona a um *pool de orders*. Os principais estados de uma *order* são *OPEN* – não processada, *PENDING* – enviada para outro membro da federação, *FULFILLED* – recurso criado e disponível, e *CLOSED* – encerrada. O controle de admissão (Fig. 1) pode ser executado quando um usuário requisita algum recurso a sua nuvem, ocasião em que uma *order* local no estado *OPEN* é adicionada ao *pool de orders* do agente, ou quando uma nuvem participante solicita re-

cursos a outro membro da federação, o que aciona o *plugin* de Controle de Capacidades[1].

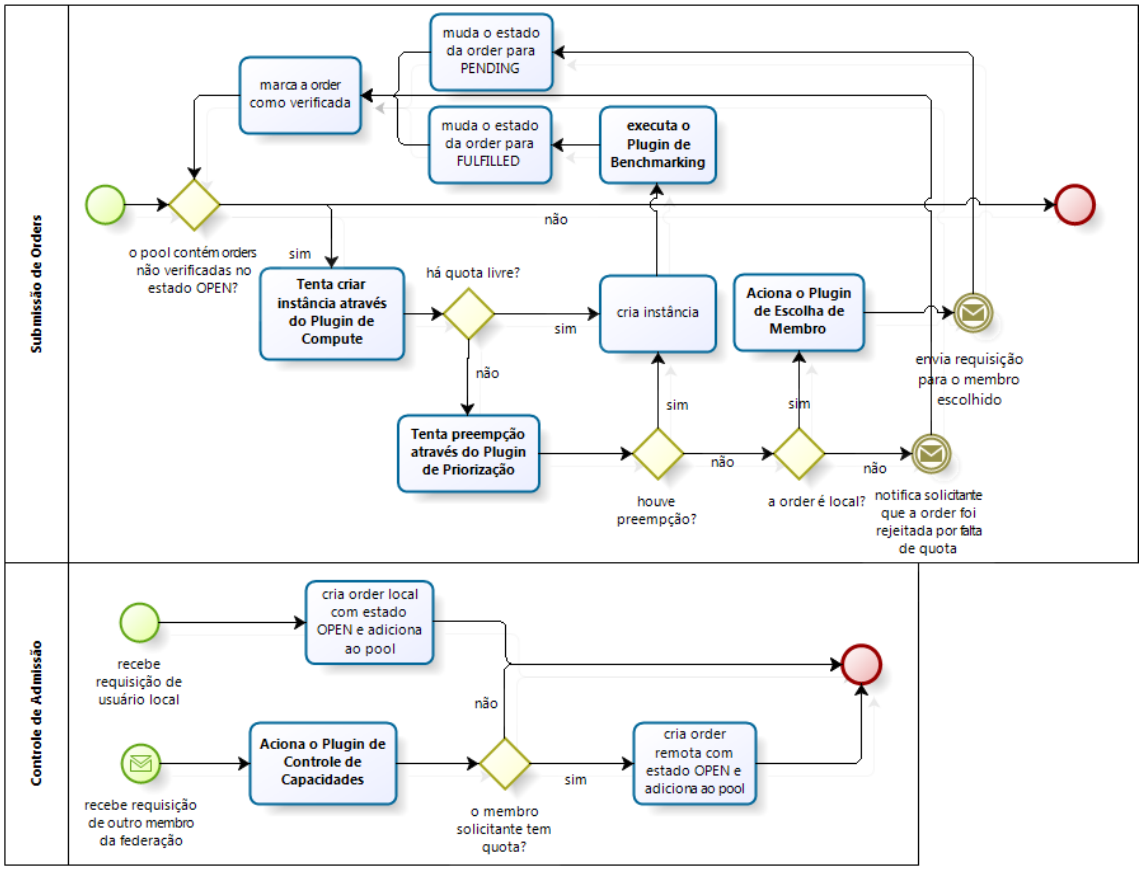

Figura 1. Processo de criação e alocação de orders.

O ***plugin* de Controle de Capacidades** atualiza periodicamente as quotas dos demais participantes e é acionado para decidir se existe quota disponível para o membro solicitante. Basicamente, o agente admitirá uma *order* remota se a quantidade de *orders* servidas, *i.e.*, *orders* no estado *FULFILLED* cujo provedor é o próprio agente, é maior ou igual à quota calculada pelo *plugin*.

O processo de submissão de *orders* (Fig. 1) também é executado periodicamente, independentemente da admissão de novas *orders*. Enquanto existir *orders* não verificadas no estado *OPEN*, o agente escolhe uma *order* segundo uma determinada sequência — *e.g.* baseado no tempo de chegada — e tenta alocar recursos localmente através do *plugin* de Compute – responsável por toda comunicação com a nuvem. Caso haja quota disponível, a nuvem subjacente cria o recurso requisitado e envia sua localização ao *plugin* de Benchmarking, responsável por medir o desempenho do recurso, para fins de contabilidade. Após a execução do *benchmarking*, o estado da *order* é configurado como *FULFILLED*, e a *order* é marcada como verificada, para que o agente não tente alocar uma mesma *order* mais de uma vez. Se a nuvem não possuir quota livre, o ***plugin* de Priorização** é acionado para verificar se há alguma *order* com menor prioridade a ser preemptada. Se alguma preempção acontecer, o recurso é criado, o *benchmarking* é executado, e o estado

[1]Os processos de autenticação e autorização foram omitidos por não ter implicação direta no escambo, e podem ser consultados em outro lugar [Brasileiro et al. 2016].

da *order* é atualizado para *FULFILLED*. Caso contrário, se a *order* não for local, o nó solicitante é notificado de que a *order* foi rejeitada por falta de quota, permitindo-o explorar o restante da federação. A *order* é marcada como verificada, de modo que seja considerada para alocação a posteriori. Caso não haja preempção, mas a *order* seja local, o agente aciona o ***plugin* de Escolha de Membro** e envia a requisição para o nó escolhido, o que aciona o controle de admissão em um outro membro da federação, e por fim, altera o estado da *order* para *PENDING*.

Para funcionamento da FD-NoF, foram implementados os seguintes plugins: i) *NoFPrioritizationPlugin*: priorização de *orders* através do sistema de saldos da NoF, em que membros com maiores saldos recebem maior prioridade em suas solicitações; ii) *FairnessDrivenCapacityControllerPlugin*: calcula as quotas dos demais membros de acordo com a abordagem FD (cf. Seção 3); iii) *RandomizedNoFMemberPickerPlugin*: heurística que prioriza a solicitação de recursos aos membros com maior débito na NoF, visto que esses nós estariam mais propensos a colaborar com participantes que lhes são recíprocos. Todas as configurações de *plugins*, tempos de execução da *thread* de submissão de *orders* e atualização das quotas, bem como os parâmetros do controlador de capacidades ($\mathcal{L}_{min}, \mathcal{L}_{max}, \Delta$) são escolhidos de forma autônoma por cada participante da federação.

5. Avaliação de Desempenho

O principal objetivo deste trabalho é avaliar o desempenho geral — níveis de paridade e satisfação — dos membros de uma federação que usa a FD-NoF como modelo de negócio. Com o propósito de obter resultados mais realistas, a FD-NoF foi implementada em um *middleware*[2] para federação de nuvens, o que remove a maioria das simplificações existentes no simulador previamente utilizado para avaliar a FD-NoF [Falcão et al. 2016a].

5.1. Infraestrutura e Simplificações

Para viabilizar cenários com intensa carga de trabalho, alguns detalhes de infraestrutura foram simplificados, mas sem que os resultados sejam comprometidos. Nos nossos experimentos a nuvem subjacente é emulada por uma classe com implementação simples do *plugin* de Compute, uma vez que a complexidade e tempo envolvidos na criação e término de máquinas virtuais (VMs) não é trivial. Além disto, para facilitar a análise dos resultados, todas as VMs solicitadas possuem os mesmos requisitos computacionais. Por esta razão, o *plugin* de Benchmarking também tem funcionamento simplificado, retornando o mesmo valor referente ao poder de processamento das VMs. Por fim, para facilitar a implantação, embora troquem mensagens normalmente via rede, todos os nós (cada um composto por um agente Fogbow – programa Java) são implantados em uma mesma máquina (20GB RAM, 16 vcpus, 500GB HD), o que nos levou a usar banco de dados na memória principal, pois o processamento de toda a carga de trabalho da federação em apenas uma máquina causaria gargalos nas operações de entrada e saída.

5.2. Método de Coleta dos Resultados

As métricas avaliadas no experimento são os níveis de paridade e satisfação de cada nó. Para isto, foi preciso monitorar em cada nó P_i a quantidade de *orders* no estado *OPEN*, *PENDING* e *FULFILLED*. Uma vez que os nós podem ter *orders* com origem local ou remota, as seguintes notações são usadas para permitir esta identificação:

[2]Disponível publicamente em `https://github.com/eduardolfalcao/fogbow-manager`.

- $O_{r=i}$: número de *orders* no estado *OPEN* em que o requerente é P_i;
- $P_{r=i}$: número de *orders* no estado *PENDING* em que o requerente é P_i;
- $F_{r=i \wedge p \neq i}$: número de *orders* no estado *FULFILLED* em que o requerente é P_i e o provedor não é P_i;
- $F_{r=i \wedge p=i}$: número de *orders* no estado *FULFILLED* em que o requerente é P_i e o provedor também é P_i; e
- $F_{r \neq i \wedge p=i}$: número de *orders* no estado *FULFILLED* em que o requerente não é P_i e o provedor é P_i.

Foi necessário registrar nos *logs* variáveis derivadas das variáveis acima. Isso acontece toda vez que uma dessas variáveis derivadas, explicadas abaixo, muda de valor. Quando isso ocorre, uma nova entrada x no *log* é registrada, com $T[x]$ igual ao tempo atual e os valores das variáveis derivadas calculados da seguinte forma:

- demanda total de P_i por unidade de tempo, entre x e $x+1$: $D_i^T[x] = O_{r=i} + P_{r=i} + F_{r=i \wedge p=i} + F_{r=i \wedge p \neq i}$;
- demanda de P_i à federação por unidade de tempo entre x e $x+1$: $D_i^F[x] = max(0, D_i^T[x] - \mathcal{C}_i)$, onde $\mathcal{C}_i$ é a capacidade máxima de P_i;
- total de recursos da federação recebidos por P_i por unidade de tempo entre x e $x+1$: $R_i^F[x] = F_{r=i \wedge p \neq i}$;
- total de recursos ofertados por P_i à federação por unidade de tempo entre x e $x+1$: $O_i^F[x] = \mathcal{C}_i - F_{r=i \wedge p=i}$;
- total de recursos de fato providos por P_i à federação por unidade de tempo entre x e $x+1$: $S_i^F[x] = F_{r \neq i \wedge p=i}$;
- total de recursos locais recebidos por P_i por unidade de tempo entre x e $x+1$: $R_i^L[x] = F_{r=i \wedge p=i}$; e
- demanda não atendida (*unattended*) pela federação por unidade de tempo, entre x e $x+1$: $U_i^F[x] = D_i^T[x] - R_i^L[x] - R_i^F[x]$.

Portanto, a paridade e satisfação até o tempo imediatamente anterior a $T[x]$ são calculadas da seguinte forma:

$$\phi_i[x] = \begin{cases} \frac{\sum_{j=1}^{x-1}(T[j+1]-T[j]) \cdot R_i^F[j]}{\sum_{j=1}^{x-1}(T[j+1]-T[j]) \cdot S_i^F[j]}, & \sum_{j=1}^{x-1} S_i^F[j] > 0 \\ \text{indefinida}, & \text{caso contrário;} \end{cases}$$

$$\psi_i[x] = \begin{cases} 1 - \frac{\sum_{j=1}^{x-1}(T[j+1]-T[j]) \cdot U_i^F[j]}{\sum_{j=1}^{x-1}(T[j+1]-T[j]) \cdot (U_i^F[j]+R_i^F[j])}, & \sum_{j=1}^{x-1} U_i^F[j] + R_i^F[j] > 0 \\ \text{indefinida}, & \text{caso contrário.} \end{cases}$$

A satisfação é calculada com uma abordagem inversa: a diferença entre 1 e a porcentagem de *orders* não atendidas. Se fosse calculada do modo mais simples, razão entre recursos atendidos e requisitados, em alguns momentos a satisfação poderia assumir valores maiores que 1. Isto poderia acontecer pois é possível que um nó receba recursos da federação enquanto possui recursos locais ociosos e, por isso, ele poderia consumir mais recursos da federação do que necessitava (levando em consideração sua capacidade local ociosa).

5.3. Cenários

O *nível de contenção* (κ, razão entre quantidade de recursos requisitados e ofertados), a estratégia da *Rede de Favores* e a presença ou ausência de *caronas* são os principais fatores a serem variados para gerar cenários de interesse.

O primeiro passo é entender quais fatores precisam ser variados, e como devem ser variados, para produzir os valores desejados de κ. Uma forma de controlar a contenção do sistema é gerando uma carga de trabalho na qual uma fração dos nós cooperativos, denotada por μ, estaria apenas doando recursos, e o restante $(1 - \mu)$ exclusivamente demandando recursos, em qualquer momento do experimento. Chamemos o intervalo de tempo que um nó deve permanecer em um dado estado (provedor ou consumidor) de λ. Portanto, para produzir um determinado valor para κ, basta gerar uma carga de trabalho que contenha a demanda total de recursos ($\mathcal{D}$) de cada nó em função de sua capacidade total de recursos ($\mathcal{C}$). Para gerar a contenção κ desejada basta configurar os nós consumidores com uma demanda total de recursos $\mathcal{D} = \mathcal{C} + (\kappa \cdot \mathcal{C})$, onde $\mathcal{C}$ recursos são satisfeitos localmente, e $\kappa \cdot \mathcal{C}$ representa a quantidade de recursos requisitados à federação. Note que a dinâmica do sistema pode levar um nó que em um dado intervalo de tempo (λ) deveria apenas ofertar recursos a também demandar recursos remanescentes do ciclo anterior. Dito isto, quanto menor for λ, maior será a probabilidade de acúmulo das demandas em ciclos posteriores, elevando a contenção do sistema. Por isso, λ é fixo em um valor intermediário de 600s, e por questões de simplificação, o valor da fração é fixo em $\mu = 0.5$.

Os dois primeiros cenários executados são configurados com contenção de recursos baixa ($\kappa < 1$, *e.g.* $\mathbb{E}[\kappa] = 0.5$) e moderada ($\mathbb{E}[\kappa] = 1$). Tais valores de contenção são apenas um limite mínimo, pois fatores como preempções e o tempo que um nó consumidor leva para encontrar um nó provedor disposto a doar recursos, pode elevar a contenção ao longo do experimento. Nestes dois cenários, todos os participantes serão configurados com capacidade total de recursos $\mathcal{C} = 20$. Sempre que estiver em estado provedor, o nó doará todos os seus recursos, e portanto $\mathcal{D} = 0$. Quando um nó estiver em estado consumidor, para gerar um nível de contenção $\mathbb{E}[\kappa] = 0.5$, a demanda precisa ser computada por $\mathcal{D} = \mathcal{C} + \frac{\mathcal{C}}{2}$, e para produzir $\mathbb{E}[\kappa] = 1$ a demanda deve ser calculada através da equação $\mathcal{D} = 2 \cdot \mathcal{C}$. Portanto, os participantes em estado consumidor serão configurados com $\mathcal{D} = 30$ para $\mathbb{E}[\kappa] = 0.5$, e $\mathcal{D} = 40$ para $\mathbb{E}[\kappa] = 1$.

Para todos os cenários, a comunidade é composta por $n \cdot (1+f)$ nós, onde n nós são colaboradores e $f.n$ são caronas. Primeiramente é avaliado o desempenho da federação sob contenção de recursos baixa e moderada, com 40 participantes cooperativos ($n = 40$ e $f = 0$), contrapondo os resultados da SD-NoF e FD-NoF ($\Delta = 0.05 \cdot \mathcal{C}$, $\mathcal{L}_{min} = 0.75$, $\mathcal{L}_{max} = 0.95$ — configuração indicada por Falcão *et. al.* (2016) por apresentar bom desempenho em cenários com diferentes contenções e porcentagem de caronas). Em seguida, o mesmo cenário de baixa contenção é reavaliado na presença de caronas, visto que é neste tipo de situação que eles conseguem se aproveitar da federação. Para isto, são adicionados 10 caronas à federação ($n = 40$ e $f = 0.25$) configurados com $\mathcal{C} = 0$ e $\mathcal{D} = 40$, *i.e.*, eles demandam coletivamente todos os recursos ofertados à federação.

6. Resultados e Discussão

O primeiro cenário é composto por 40 nós, todos cooperativos, configurados com $\mathcal{C} = 20$ e $\mathcal{D} \in \{0, 30\}$ para os estados provedor e consumidor do cenário com $\mathbb{E}[\kappa] = 0.5$, e

$\mathcal{D} \in \{0, 40\}$ para o cenário com $\mathbb{E}[\kappa] = 1$. O experimento tem duração de 24 horas, e os nós alternam entre consumidor e provedor a cada $\lambda = 600$ segundos. Cada cenário foi executado com 6 replicações. A Figura 2 apresenta os valores mínimo, primeiro quartil, mediana, média, terceiro quartil, 99.48 percentil e máximo para os níveis de paridade, e os valores mínimo, percentil 0.42, primeiro quartil, mediana, média, terceiro quartil e máximo de satisfação dos nós, nas 6 replicações, ao longo do experimento.

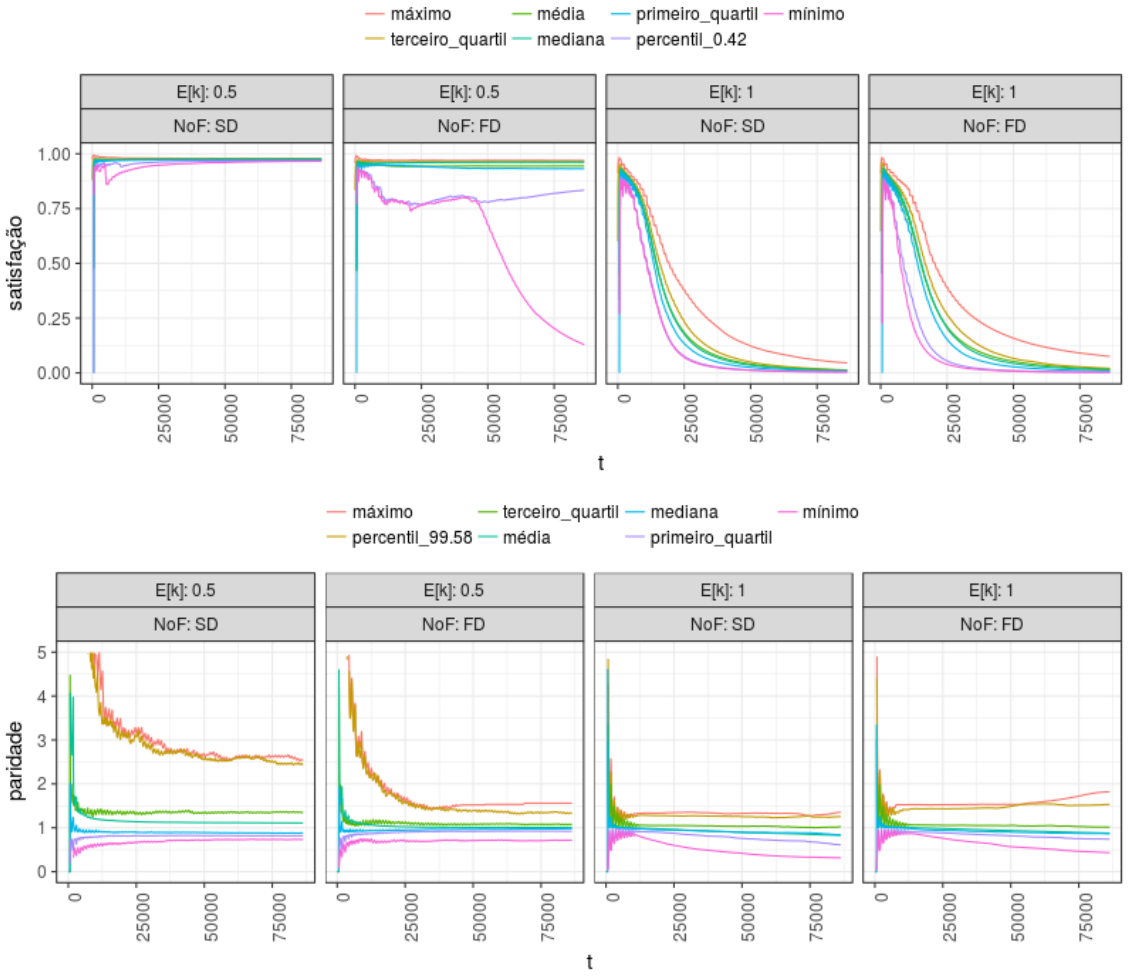

Figura 2. Níveis de paridade e satisfação dos participantes na SD-NoF e FD-NoF, no cenário sem caronas e com $\mathbb{E}[\kappa] \in \{0.5, 1\}$.

A partir da Figura 2 é possível observar que a FD-NoF teve desempenho semelhante à SD-NoF, o que é bom, visto que na ausência de caronas faz sentido ofertar todos os recursos ociosos. Em suma, ao monitorar a paridade do nó com os demais membros, o controlador percebe que eles são recíprocos e por isso mantém a oferta em um alto patamar. No entanto, pode-se notar que um nó teve desempenho atípico (vide gráfico de satisfação com FD-NoF e $\mathbb{E}[\kappa] = 0.5$). Isto aconteceu pois, a longo prazo, este nó retribuiu aproximadamente apenas metade dos recursos recebidos da federação, o que lhe rendeu ao fim do experimento uma paridade de 1.56. Como neste cenário $\mathbb{E}[\kappa] = 0.5$, mais recursos são ofertados do que demandados e, com isto, um provedor pode não conseguir doar todos os seus recursos ociosos na mesma proporção que receber, aparentando ser um carona. Apesar de improvável, isto aconteceu com 1 ($\approx 0.42\%$) dos 240 nós.

Contudo, ao contrário do cenário com SD-NoF e $\mathbb{E}[\kappa] = 0.5$, no qual a contenção real se manteve em $\kappa(t) \approx 0.5, \forall t \in [0, 86400]$, os níveis de contenção real para o cenário com SD-NoF e $\mathbb{E}[\kappa] = 1$ aumentaram consideravelmente — $\kappa(t = 28800) = 26.9$, $\kappa(t = 57600) = 65.1$ e $\kappa(t = 86400) = 88.9$ — o que atenuou os níveis de satisfação no último momento do experimento para um valor máximo de 4%. Isto aconteceu graças ao tempo elevado que os consumidores levaram para encontrar provedores com recursos disponíveis e à ocorrência de preempções, cuja frequência é maior em cenários de contenção elevada. Isto fez com que as demandas dos nós se acumulassem ao longo do experimento – situação não percebida no modelo de simulação simplificado.

Com relação aos níveis de paridade, observa-se que a FD-NoF consegue, em cenários de baixa contenção por recursos, diminuir a dispersão entre os níveis de paridade, tornando a federação mais justa. Em cenários com $\mathbb{E}[\kappa] = 0.5$, no último momento do experimento os níveis de paridade dos nós na SD-NoF e FD-NoF permaneceram, respectivamente, dentro dos intervalos $[0.73..2.53]$ e $[0.7..1.56]$. Quando $\mathbb{E}[\kappa] = 1$, 99.58% dos nós na SD-NoF obtiveram níveis de paridade entre 0.31 e 1.35, ao passo que na FD-NoF, os mesmos nós obtiveram níveis de paridade no intervalo $[0.43..1.53]$, que são patamares semelhantes, uma vez que para $\mathbb{E}[\kappa] = 1$ a SD-NoF por si só já é eficiente.

O segundo cenário é composto por 50 participantes, onde 10 são caronas. Os nós cooperativos são configurados com as mesmas demandas e capacidades do cenário anterior em que $\mathbb{E}[\kappa] = 0.5$. Os caronas, por outro lado, são configurados com $\mathcal{C} = 0$ e $\mathcal{D} = 40$, *i.e.*, sempre demandam todos os recursos ofertados à federação. O experimento tem duração de 24 horas com $\lambda = 600$ segundos, e são executadas 6 replicações para cada cenário. A Figura 3 apresenta os níveis de satisfação de cada nó, no último momento do experimento, plotados em diagramas de caixa ordenados por replicação (*i.e.*, as caixas vermelha e azul mais à esquerda exibem os resultados de uma mesma replicação), em cenários onde os nós interagem usando apenas a SD-NoF ou FD-NoF. A paridade dos nós cooperativos[3] é apresentada de modo resumido na Figura 3 através dos seguintes intervalos: mínimo, primeiro quartil, mediana, média, terceiro quartil, percentil 90 e máximo.

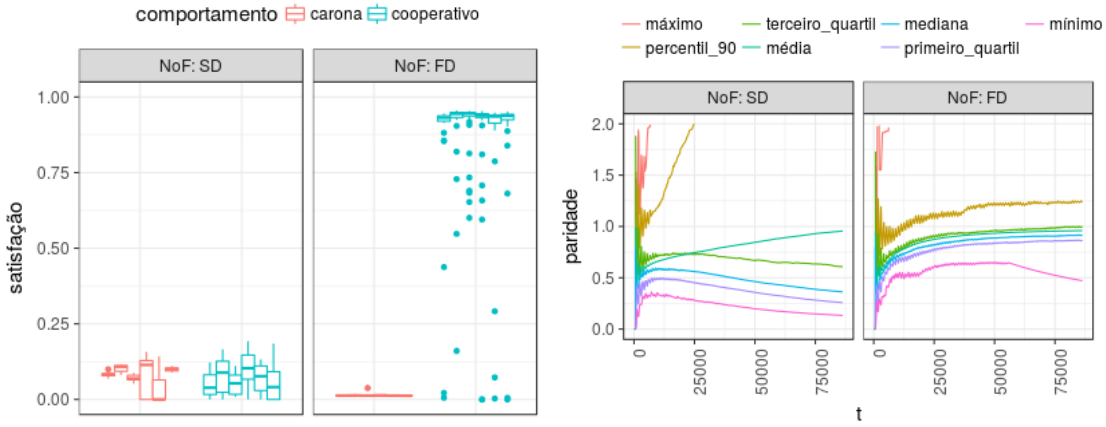

Figura 3. Níveis de satisfação (nós cooperativos e caronas) e paridade (nós cooperativos) na SD-NoF e FD-NoF, no cenário com caronas e $\mathbb{E}[\kappa] = 0.5$.

A Figura 3 torna evidente que os nós equipados com a SD-NoF não foram capazes

[3]Por definição, os caronas tem paridade igual -1, visto que nunca doam recursos.

de identificar e evitar os caronas em cenários com baixa contenção de recursos. Nas 6 replicações desses cenários, o nível de satisfação dos nós cooperativos e caronas no último momento do experimento variou em patamares semelhantes, respectivamente entre $[0.009..0.19]$ e $[0.05..0.15]$. A FD-NoF, por outro lado, assegurou altos níveis ($[0.9..0.95]$) de satisfação a 85% dos nós cooperativos enquanto todos os caronas só obtiveram de 1% à 3% de suas solicitações atendidas. No entanto, ainda na FD-NoF, alguns nós apresentaram desempenho insatisfatório (*e.g.*, 4.5% dos nós cooperativos obtiveram níveis de satisfação inferiores a 50%) por terem alcançado altos níveis de paridade e, por isto, os demais nós o perceberam como carona e cessaram as doações para eles. Adicionalmente, a FD-NoF também conseguiu diminuir a diferença de paridade em relação à SD-NoF, em que ao se considerar 90% dos nós, permaneceram respectivamente entre $[0.47..1.25]$ e $[0.13..4.02]$.

Dois fatores que têm impacto direto nos resultados são a configuração do Controlador de Capacidades e o valor de λ. É possível que menores valores para $\mathcal{L}_{min}$ e Δ e um período mais longo para execução do controlador (ao invés de 30 segundos) oferecessem mais oportunidades aos nós cooperativos que foram segregados juntamente com os caronas de retribuírem os recursos recebidos e se mostrarem recíprocos. No entanto, esta configuração também poderia favorecer os caronas, uma vez que a FD-NoF levaria mais tempo para identificá-los. Com relação ao λ, quanto menor seu valor mais difícil se torna a retribuição dos recursos pelos nós cooperativos em tempo hábil, pois os consumidores não encontram potenciais provedores de modo simples e direto como acontece em um sistema de mercado centralizado; de maneira oposta, mecanismos de escambo, por se tratarem de uma abordagem descentralizada, contam com uma morosidade

7. Considerações Finais e Trabalhos Futuros

Este trabalho descreveu detalhes de implementação da FD-NoF – um mecanismo de escambo cujo objetivo é promover reciprocidade – em um *middleware* para federação de provedores de computação na nuvem, e apresentou seu desempenho através de experimentos conduzidos em um ambiente controlado.

A experimentação em um contexto mais realista ratificou resultados previamente encontrados com o auxílio de um modelo de simulação simplificado [Falcão et al. 2016a]: a FD-NoF obtêm resultados semelhantes aos da SD-NoF em cenários de alta contenção (bons níveis de paridade), e em cenários de baixa contenção a FD-NoF consegue identificar e isolar os caronas, assegurando reciprocidade entre os participantes. Adicionalmente, foi possível ter uma noção mais aprofundada acerca das dificuldades envolvidas na busca por participantes dispostos a cooperar, e o fato de que em alguns momentos alguns participantes consomem da federação mais recursos do que de fato necessitavam.

Ainda com o mesmo nível de controle do ambiente, trabalhos futuros incluem investigar o desempenho de uma federação cuja carga de trabalho seja produzida a partir de diferentes valores para λ, e investigar o impacto de uma configuração mais agressiva para o controlador (maiores valores para Δ e $\mathcal{L}_{min}$). Também seria interessante executar experimentos com cargas de trabalho representativas, coletadas em sistemas reais.

Agradecimentos

Este trabalho foi financiado pela FAPESP (processo nº 2015/24461-2) e CAPES (processo nº 1361750). Francisco Brasileiro é pesquisador CNPq (processo nº 311297/2014-5).

Referências

[Andrade et al. 2007] Andrade, N., Brasileiro, F., Cirne, W., and Mowbray, M. (2007). Automatic grid assembly by promoting collaboration in peer-to-peer grids. *Journal of Parallel and Distributed Computing*, 67(8):957 – 966.

[Assis and Bittencourt 2016] Assis, M. and Bittencourt, L. (2016). A survey on cloud federation architectures: Identifying functional and non-functional properties. *Journal of Network and Computer Applications*, 72:51 – 71.

[Brasileiro et al. 2016] Brasileiro, F., Silva, G., Araújo, F., Nóbrega, M., Silva, I., and Rocha, G. (2016). Fogbow: A middleware for the federation of IaaS clouds. In *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 531–534.

[Cohen 2003] Cohen, B. (2003). Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72.

[Falcão et al. 2016a] Falcão, E., Brasileiro, F., Brito, A., and Vivas, J. (2016a). Enhancing fairness in p2p cloud federations. *Computers & Electrical Engineering*, 56:884 – 897.

[Falcão et al. 2015] Falcão, E., Brasileiro, F., Brito, A., and Vivas, J. L. (2015). Controlando a contenção de recursos para promover justiça em uma federação peer-to-peer de nuvens privadas. In *Anais do XIII Workshop em Clouds e Aplicações*.

[Falcão et al. 2016b] Falcão, E., Brasileiro, F., Brito, A., and Vivas, J. L. (2016b). Enhancing p2p cooperation through transitive indirect reciprocity. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*.

[Gomes et al. 2012] Gomes, E. R., Vo, Q. B., and Kowalczyk, R. (2012). Pure exchange markets for resource sharing in federated clouds. *Concurrency and Computation: Practice and Experience*, 24(9):977–991.

[Grozev and Buyya 2014] Grozev, N. and Buyya, R. (2014). Inter-cloud architectures and application brokering: taxonomy and survey. *Software: Practice and Experience*, 44(3):369–390.

[Kamvar et al. 2003] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. (2003). The eigentrust algorithm for reputation management in p2p networks. In *12th Web Conf*, pages 640–651.

[Kecskemeti et al. 2012] Kecskemeti, G., Kertesz, A., Marosi, A., and Kacsuk, P. (2012). Interoperable resource management for establishing federated clouds. *Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice*, 2:18–35.

[Kumar et al. 2011] Kumar, C., Altinkemer, K., and De, P. (2011). A mechanism for pricing and resource allocation in peer-to-peer networks. *Electronic Commerce Research and Applications*, 10(1):26 – 37. Special Section: Service Innovation in E-Commerce.

[Menasché et al. 2010] Menasché, D. S., Massoulié, L., and Towsley, D. (2010). Reciprocity and barter in peer-to-peer systems. In *Proc. of 29th INFOCOM*, pages 1505–1513.

[Petri et al. 2015] Petri, I., Diaz-Montes, J., Zou, M., Beach, T., Rana, O., and Parashar, M. (2015). Market models for federated clouds. *IEEE Transactions on Cloud Computing*, 3(3):398–410.

[Zhao and Li 2009] Zhao, H. and Li, X. (2009). H-trust: A group trust management system for peer-to-peer desktop grid. *Journal of Computer Science and Technology*, 24(5):833–843.