UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Avaliação de Esquemas de Contabilidade Autônoma em Grades Computacionais Peer-to-Peer

Robson Hugo Araújo dos Santos

Campina Grande – PB
Agosto - 2005

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Avaliação de Esquemas de Contabilidade Autônoma em Grades Computacionais Peer-to-Peer

Robson Hugo Araújo dos Santos

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal de Campina Grande - Campus I - como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Walfredo Cirne
Orientador

Campina Grande – PB
Agosto - 2005

SANTOS, Robson Hugo Araújo dos.
S231A

Avaliação de Esquemas de Contabilidade Autônoma em Grades *Peer-to-Peer***.**

Dissertação (Mestrado) – Universidade Federal de Campina Grande, Centro de Ciências e Tecnologia, Coordenação de Pós-Graduação em Informática, Campina Grande – Paraíba, Agosto/2005.

85 p. Il.

Orientador: Walfredo da Costa Cirne Filho

Palavras Chave:

1. Sistemas Distribuídos
2. Grades Computacionais
3. *Peer-to-Peer*
4. Contabilidade de Recursos

CDU-681.3.066

Agradecimentos

Antes de tudo, agradeço a Deus.

A minha família, pela preocupação e por ter contribuído em toda minha formação.

A Ana Carolina, por seu carinho e por ter me incentivado sempre e contribuído efetivamente para a realização deste trabalho.

Aos professores Elmar e Joseana, por terem me orientado durante o início do meu mestrado e por ter compreendido minha opção de mudar de grupo de pesquisa.

Ao meu Orientador e Guru Walfredo, pela amizade, pela orientação científica e filosófica e por ter acompanhado de perto meu trabalho tendo competência em guiá-lo até sua conclusão.

Ao professor Fubica, pelas discussões e contribuições fundamentais para a conclusão do trabalho.

Ao professor Nazareno (É bicho, só falta mesmo o título! ☺), pela amizade e co-orientação deste trabalho.

Ao professor Bruno Schulze, por ter aceitado avaliar e contribuir com este trabalho.

Aos meus amigos do Mestrado e do Laboratório de Sistemas Distribuídos, que contribuíram para que o meu ambiente de trabalho fosse sempre agradável. Em especial a Alisson, Gustavo, Lauro, Aliandro, Mirna, Ayla, Eliane, Érica, Esther, Andrey, Daniel Fireman, Felipe, Elizeu, Ana Cristina, Rodrigo Rebouças, Luana, Rogério, Daniel, Elthon, Emerson e Verlaynne.

A Aninha, Keka, Zane, Marcelo e Vera pelo apoio e amizade.

Ao pessoal do Residencial Flamingo (“Favela”), pelo ambiente descontraído e pela amizade. Em especial a David, meu “brother” durante a minha estadia em Campina, e a Nelson e Afrânio pelo aperreio e amizade.

Enfim, a todos que de alguma maneira ajudaram (ou não atrapalharam☺) na realização deste trabalho.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Grades computacionais peer-to-peer têm evoluído consideravelmente nos últimos anos por possibilitarem o compartilhamento de recursos computacionais entre organizações. É importante que estes sistemas estimulem os peers a disponibilizarem seus recursos ociosos, pois se não fizerem isso, os peers podem não colaborar e o sistema pode entrar em colapso. [1] introduz a Rede de Favores, uma abordagem descentralizada e baseada em reputação para resolver este problema na grade computacional peer-to-peer OurGrid. Entretanto, para a Rede de Favores ser robusta, é necessário fornecer um esquema de contabilidade que seja preciso, simples, descentralizado e resistente a ataques. Prover um mecanismo de contabilidade preciso para um sistema no qual os participantes podem se comportar de maneira maliciosa não é trivial. Neste trabalho, propomos o esquema de contabilidade por poder relativo do peer, um esquema que contabiliza o uso de recursos de maneira totalmente autônoma, isto é, baseado somente em informações que um peer pode computar localmente. A autonomia da contabilidade torna desnecessária a existência de relações de confiança entre os peers. O poder relativo de um peer remoto é uma estimativa de quão rápido ele é em relação ao peer local. O uso dos recursos pelo peer remoto é calculado como o produto do poder relativo desse peer e o tempo no qual o recurso esteve disponível.

Para validar nossa solução, desenvolvemos um simulador para grades computacionais peer-to-peer baseado em eventos. Com a ajuda desse simulador, avaliamos o esquema de contabilidade por poder relativo e o comparamos com o esquema de contabilidade perfeita, um esquema de contabilidade alimentado com informações perfeitas sobre o custo computacional das tarefas das aplicações e o poder computacional dos recursos (que não é possível de se implementar na prática para os nossos sistemas alvos) e um esquema de contabilidade por tempo, esquema de contabilidade autônoma em que o consumidor e o provedor de recursos medem apenas o tempo no qual o recurso esteve disponível. Os nossos resultados mostram que a contabilidade por poder relativo é mais precisa do que a contabilidade por tempo para as métricas de razão de favores e tempo de resposta das aplicações. Além disso, a precisão da contabilidade por poder relativo é muito próxima da do esquema de contabilidade perfeita.

Abstract

Peer-to-peer computational grids have evolved considerably in late years because they allow the sharing of computational resource among organizations. It is important that these systems provide incentives for peers to offer their idle resources. If they do not do so, peers may not want collaborate and the system may collapse. [1] presents the Network of Favors, a decentralized approach based on reputation to solve this problem in the peer-to-peer computational grid OurGrid. However, in order for the Network of Favors to be robust, it is necessary to provide an accounting scheme that is accurate, simple, decentralized and resistant to attacks. Providing an accurate mechanism of accounting for a system in which the parties can have a malicious behavior is not trivial. In this work, we propose the accounting scheme based on relative power of peers, a scheme that accounts the use of resources in a totally autonomous way, that is, only based on information that a local peer can compute. The autonomy of the accounting makes the existence of trust relationships among peers unnecessary. The relative power of a remote peer is an estimate of how fast it is in relation to another. The use of resources by another peer is calculated as a product of the relative power of this peer versus the time in which the resource is available.

In order to validate our solution we developed an event-based simulator for peer-to-peer computational grids. With the aid of this simulator, we evaluated the accounting scheme by relative power and compared it with the perfect accounting scheme, an accounting schema fed by perfect information about the computational cost of the application tasks and the computational power of resources - which is not possible to implement in practice for our target systems – and an accounting schema by time, an autonomous accounting schema in which a consumer and a provider of resources measure only the time of the availability of the resource. Our results show that the accounting by relative power is more accurate than the accounting by time for the metrics favor ratio and application response time. In addition to this, the accuracy of the accounting by relative power is very near to the one of the perfect accounting schema.

CAPÍTULO 1

Introdução

Este capítulo apresenta a motivação para o trabalho descrito nesta dissertação. Primeiramente, é mostrada uma contextualização da área de computação em grade e sistemas peer-to-peer. Em seguida, é apresentada a definição do problema atacado nesta dissertação: a contabilização da utilização de recursos em grades peer-to-peer e discutida sua relevância. Ao final, mostramos a estrutura da dissertação com uma breve descrição dos capítulos.

1.1 Contextualização

Grades computacionais são ambientes de compartilhamento, implementados através da implantação de uma infra-estrutura persistente que suporta a criação de comunidades distribuídas e o compartilhamento de recursos nestas comunidades [18]. [18] define peer-to-peer como a classe de aplicações que utiliza recursos, tais como, área de armazenamento, ciclos computacionais, conteúdo, presença humana, disponível na Internet. Os sistemas peer-to-peer geralmente são descentralizados e auto-organizáveis e a maioria da comunicação envolvida é simétrica.

As grades computacionais e sistemas peer-to-peer permitem a criação de comunidades pela junção de recursos em vários domínios administrativos. Geralmente, as grades computacionais agregam uma grande quantidade de recursos de poucos domínios administrativos, com poucas restrições na confiança entre as partes, e provêem muitos serviços, com exigências não triviais na qualidade de serviço [18]. Já os sistemas peer-to-peer lidam com uma grande quantidade de participantes, porém, oferecem serviços limitados e especializados, com pouca (ou nenhuma) restrição de qualidade serviço e confiança entre as partes. Neste contexto, definimos grades computacionais peer-to-peer como a infra-estrutura persistente que utiliza sistemas peer-to-peer para o compartilhamento de poder computacional em larga escala na qual todos os peers podem ser consumidores e provedores de recursos.

A computação em grade e os sistemas peer-to-peer vêm para suprir a crescente necessidade de recursos decorrente do aumento da complexidade de aplicações científicas e comerciais e da quantidade de dados que precisam ser processados por estas aplicações. Por ser uma tecnologia mais barata do que as plataformas paralelas

tradicionais, pois não demanda custo de manutenção de todos os recursos necessários para executar as aplicações, até organizações pequenas ou médias, como a maioria dos laboratórios de pesquisa, podem se beneficiar desta tecnologia para a execução de aplicações que demandem, por exemplo, grande quantidade de poder computacional [19].

Como os recursos disponíveis em grades computacionais são finitos e há um custo para prover recursos na grade, é importante incentivar os provedores de recursos da grade a tornarem seus recursos disponíveis. Existem duas alternativas principais neste sentido: seguir uma economia de grade [9] ou utilizar um esquema baseado em reputação [1].

Uma economia de grade usa modelos baseados em mercado para definir quem terá acesso aos recursos da comunidade. Implementações desses modelos utilizam esquemas baseados em oferta e demanda tais como, leilão, contratos e catálogo de preços para priorizar as alocações de recursos [9]. Oferta e demanda também determinam os preços para utilizar os recursos. Esses preços podem estar baseados em moedas virtuais ou mesmo em moedas reais ou em garantias de que o dono do recurso obterá recursos em troca quando precisar. A principal vantagem das economias de grade é que os modelos utilizados por ela já existem em sistemas não computacionais, e, portanto é necessário apenas adaptá-los para a utilização em ambientes computacionais. Por outro lado, a desvantagem é que a infra-estrutura necessária, como, por exemplo, sistemas de moeda eletrônica e sistemas gerais para auditoria de serviços, ainda não está amplamente disponível.

Modelos baseados em reputação utilizam o comportamento dos participantes no sistema para determinar as prioridades na alocação de recursos. Deste modo, enquanto os participantes contribuem com recursos para a comunidade, suas chances de obter recursos de volta da comunidade aumentam. A maior vantagem dessa abordagem é a simplicidade de implantação de tais modelos, que podem ser totalmente distribuídos, como por exemplo, na Rede de Favores [1]. Por outro lado, nestas soluções existem poucas garantias de qualidade de serviço. Além disso, é necessário incentivar os provedores a disponibilizarem os recursos no sistema. Pois nesses sistemas, há a necessidade de evitar que *free-riders,* isto é, participantes que não contribuem para a comunidade de forma proporcional ao consumo deles, tenham a mesma prioridade no acesso aos recursos que os usuários que contribuem para a comunidade.

1.2 Definição do problema

As organizações, especialmente as comerciais, não têm incentivos para contribuir em comunidades de compartilhamento de recursos, a menos que sejam "pagas" para prover os recursos delas. Para possibilitar esse pagamento pelo uso dos recursos, é necessário avaliar a utilização dos recursos pelos usuários da comunidade e manter registro dela. Esse é o objetivo principal de sistemas de contabilidade.

A implantação de sistemas de contabilidade de recursos em ambientes distribuídos não é uma tarefa fácil. Em grades computacionais peer-to-peer esta dificuldade é acentuada devido às entidades do sistema estarem espalhadas em domínios administrativos diferentes. Estar em domínios administrativos diferentes significa que um participante não tem acesso completo aos recursos de outro participante e que pode não haver confiança mútua entre os participantes da comunidade. Assim, dois grandes problemas enfrentados para contabilizar recursos em grades peer-to-peer são encontrar uma boa métrica de contabilidade e a inexistência de confiança entre os participantes da comunidade.

A dificuldade em encontrar uma métrica para contabilidade é definir o quanto um recurso é útil para cada usuário da grade computacional, ou seja, é necessário encontrar um valor que mostre o tamanho da contribuição de um recurso para a execução de uma aplicação submetida à grade computacional. Uma possível métrica poderia ser a quantidade de operações de ponto flutuante realizadas pelo recurso. Porém, a quantidade de operações de ponto flutuante varia dependendo da plataforma de execução. Uma alternativa pode ser utilizar como métrica apenas o tempo gasto pelo recurso para processar a tarefa. Porém, como iremos mostrar no capítulo 4 e no capítulo 5, essa métrica não reflete a quantidade de trabalho que cada recurso pode realizar.

Além do uso de CPU, um esquema de contabilidade deve refletir também outros fatores necessários para executar as tarefas, visto que para algumas aplicações a largura de banda da rede e a velocidade dos dispositivos de entrada e saída são mais importantes que o poder de processamento. Por exemplo, deve-se levar em conta o tempo de acesso necessário para utilizar um recurso em um domínio administrativo diferente. Nesse caso, recursos com alta capacidade de processamento podem ser menos valiosos se estiverem em uma rede cujo acesso seja alto em relação a recursos com

menor capacidade de processamento com um custo de acesso menor para uma aplicação que necessita transferir uma grande massa de dados.

Para o consumidor, acreditar no provedor de recursos pode facilitar a contabilidade uma vez que os provedores têm conhecimento sobre os recursos e acesso completo aos mesmos. Entretanto, os consumidores não podem confiar nessa contabilidade, uma vez que os peers em uma grade peer-to-peer não necessariamente têm relações de confiança entre si e existe um incentivo econômico para que provedores de recursos fraudem a contabilidade informada para simular uma contribuição maior do que a efetivamente realizada. Então, os consumidores precisam de uma maneira de verificar a contabilidade fornecida ou realizar a contabilidade de forma independente.

1.3 Objetivos

O principal objetivo desta dissertação é avaliar esquemas de contabilidade autônoma para grades computacionais peer-to-peer em que não exista confiança mútua entre os participantes. Para atingir o objetivo geral, seguem os seguintes objetivos específicos:

- Identificar e propor esquemas autônomos para contabilizar a utilização de recursos em grades computacionais peer-to-peer;
- Desenvolver um simulador de grades computacionais peer-to-peer para avaliar os esquemas de contabilidade;
- Identificar métricas e cenários de simulação para avaliar os esquemas de contabilidade propostos;
- Comparar os esquemas de contabilidade propostos com um esquema de contabilidade referência, que definimos como um esquema de contabilidade alimentado com informações perfeitas sobre os recursos e as aplicações. Note que o esquema perfeito serve apenas como base de comparação, não podendo ser efetivamente implementado nos sistemas que consideramos.

1.4 Contribuições

Nesta dissertação avaliamos dois esquemas de contabilidade: a contabilidade por tempo e a contabilidade por poder relativo.

Os resultados dos esquemas de contabilidade foram comparados com o esquema de contabilidade perfeita, um esquema de contabilidade alimentado com informações perfeitas sobre o custo computacional das tarefas das aplicações e o poder computacional dos recursos, o que torna este esquema inviável de ser implementado na prática para os nossos sistemas alvos.

A contabilidade por tempo é a forma mais simples de fazer contabilidade autônoma em grades peer-to-peer, pois o tempo é uma informação fácil de ser mensurada tanto para o provedor de recursos quanto para o consumidor. Porém, como mostramos no Capítulo 4 e no Capítulo 5, esse esquema de contabilidade não incentiva os provedores de recursos a disponibilizarem recursos rápidos na grade. Além disso, ele não é resistente a um ataque simples no qual o provedor executa mais de uma tarefa no mesmo recurso ao mesmo tempo, aparentando para a grade ter mais recursos do que realmente tem.

Com o objetivo de realizar a contabilidade de recursos de forma mais robusta, mas mantendo a simplicidade de implantação, propomos nesta dissertação um novo esquema de contabilidade, a contabilidade por poder relativo. Esse esquema utiliza apenas informações que o peer consegue computar localmente para realizar contabilidade de recursos. Para mensurar o quão úteis para a computação são as contribuições de um peer remoto, calculamos o poder relativo do peer. Essa métrica fornece uma estimativa de quão rápido são, em média, os recursos de um peer remoto em relação aos recursos locais. O cálculo do poder relativo é realizado dividindo o tempo médio em que as tarefas de uma aplicação submetida ao peer consumidor demoram em executar nos recursos locais pelo tempo em que as tarefas da mesma aplicação demoram em executar no peer provedor (Ver Capítulo 4). O uso dos recursos pelo peer remoto é calculado como o produto do poder relativo desse peer e o tempo no qual o recurso esteve disponível. A autonomia da contabilidade torna desnecessária a existência de relações de confianças entre os peers, visto que não é necessário que os peers troquem informação para a realização da contabilidade.

Para validar nossa solução, desenvolvemos um simulador para grades computacionais peer-to-peer baseado em eventos. Esse simulador modela os principais aspectos envolvidos na execução de aplicações em grades computacionais peer-to-peer. Mais especificamente, esse simulador foi baseado no modelo utilizado pelo OurGrid

[1]. Uma contribuição desse simulador é que ele pode ser reutilizado em pesquisas que envolvam o desenvolvimento de soluções para o OurGrid.

Os nossos resultados mostram que a contabilidade por poder relativo apresenta resultados mais próximos da contabilidade perfeita do que a contabilidade por tempo para as nossas métricas: razão de favores e tempo de resposta das aplicações. A contabilidade por tempo é precisa apenas em cenários onde os recursos são homogêneos, ou seja, quando potência média das máquinas dos peers não apresenta variações. Porém, não é precisa em cenários em que a potência média dos peers varia ou os peers atacam a contabilidade executando mais de uma tarefa simultaneamente em cada recurso. A contabilidade por poder relativo se mostrou precisa nos cenários avaliados, inclusive, quando variamos o custo computacional das tarefas das aplicações submetidas.

1.5 Estrutura da Dissertação

O texto desta dissertação está dividido em seis capítulos.

O Capítulo 2 define os principais conceitos necessários para o entendimento desta dissertação. Também são apresentados como alguns sistemas peer-to-peer e de computação em grade contabilizam recursos.

O Capítulo 3 apresenta a metodologia utilizada para a avaliação dos esquemas de contabilidade. Nesse capítulo, definimos o modelo de grade considerado nesta dissertação, o simulador que implementa este modelo, os cenários nos quais avaliamos os esquemas de contabilidade e as métricas consideradas.

O Capítulo 4 define a contabilidade por tempo, um esquema de contabilidade que utiliza apenas o tempo no qual o recurso ficou disponível para o consumidor como contabilidade, tanto do lado do provedor quanto do lado do consumidor. Neste capítulo também detalhamos os principais aspectos do esquema de contabilidade por poder relativo, mostrando um exemplo de uso.

O Capítulo 5 apresenta a avaliação dos esquemas de contabilidade autônoma nos cenários definidos no Capítulo 3. A ênfase da discussão são os pontos fortes e fracos de cada esquema de contabilidade e a comparação deles com o esquema de contabilidade perfeita.

O Capítulo 6 traz conclusões sobre os resultados alcançados neste trabalho e apresenta sugestões para trabalhos futuros.

Por fim, o Apêndice A, apresenta um artigo técnico sobre contabilidade de recursos em grades computacionais.

CAPÍTULO 2

Estado da Arte

Sistemas distribuídos permitem a distribuição da computação das aplicações sobre vários sistemas computacionais. Atualmente, existe um grande número de pesquisas nas áreas de computação em grade e sistemas peer-to-peer visto que estas tecnologias diminuem os custos das instituições com aquisição e manutenção de recursos devido a sua a capacidade de compartilhar recursos entre domínios administrativos diferentes que essas duas tecnologias propiciam. Os recursos compartilhados por esses sistemas são largamente heterogêneos, desde computadores pessoais e estações de trabalho a supercomputadores e dispositivos especiais de visualização, passando por instrumentos digitais.

Alguns dos objetivos desses sistemas são: (i) utilizar um conjunto de recursos como, por exemplo, computadores, redes e sistemas de armazenamento, para resolver problemas que um único computador não consegue resolver, (ii) aproveitar os ciclos ociosos de computadores, (iii) permitir o acesso sob demanda a recursos computacionais e (iv) compartilhar dados entre organizações.

Para a consecução desses objetivos, em uma grade peer-to-peer, cada peer representa um domínio administrativo diferente. Esses peers se juntam e formam uma comunidade de compartilhamento de recursos. Cada peer controla o acesso a um conjunto de recursos locais e recebe requisições de usuários locais. Nessa comunidade, os peers podem fazer o papel de consumidores e/ou provedores ao longo do tempo. Um peer atua como provedor quando atende requisições de recursos oriundas da comunidade e disponibiliza seus recursos livres para responder à requisição. Um peer atua como consumidor quando utiliza seus próprios recursos ou os recursos livres disponibilizados por outros peers da comunidade.

Quanto à distribuição, classificamos os esquemas de contabilidade como centralizados, distribuídos e autônomos. A contabilidade é centralizada quando é realizada por uma única entidade na grade. A contabilidade é distribuída quando é realizada por diversas entidades na grade que trocam informações para gerar a contabilidade de uso de recursos. O objetivo deste trabalho é realizar contabilidade autônoma, isto é, contabilidade realizada por cada participante sem que haja troca de informações. Nesse caso, os valores contabilizados são mantidos por cada peer e não

existe necessidade de acordo entre provedor e consumidor para o valor do serviço, ou seja, o consumidor e o provedor de recursos podem registrar valores diferentes para a utilização de um mesmo serviço. A motivação para esse tipo de contabilidade é que é difícil manter um ponto único para realizar contabilidade em ambientes distribuídos como grades computacionais e sistemas peer-to-peer, pois exige esforço em infra-estrutura, como por exemplo, auditoria e criptografia. Além disso, nem sempre existe confiança mútua entre os peers ou confiança em uma terceira parte.

Neste capítulo vamos mostrar as principais características da grade computacional peer-to-peer OurGrid. Em seguida, descreveremos alguns modelos econômicos propostos para grades computacionais. Após isso, mostraremos alguns modelos e sistemas de contabilidade para grades computacionais.

2.1 Comunidade OurGrid

A comunidade OurGrid é uma grade computacional peer-to-peer que visa a solucionar os problemas de estabelecimento da grade para usuários de aplicações bag-of-tasks, aplicações nas quais as tarefas não precisam de comunicação para serem executadas. Mesmo sendo destinada apenas para aplicações bag-of-tasks, a solução OurGrid é útil na resolução de uma grande variedade de problemas tais como mineração de dados, simulações, cálculos fractais e processamento de imagens [5] [8].

Uma característica importante da comunidade OurGrid é prover acesso a recursos que os usuários não conseguem acessar diretamente, em outras palavras, recursos nos quais os usuários não têm login/senha. Neste sentido, o OurGrid funciona como um provedor para os recursos locais (recursos que estão no mesmo domínio administrativo) e recursos remotos (recursos dos outros peers). Os usuários de peers externos podem utilizar os recursos providos até que algum usuário do peer local precise utilizá-lo ou até que suas aplicações sejam concluídas.

Para decidir para quem os recursos devem ser alocados quando existe contenção de recursos, o OurGrid utiliza o esquema de rede de favores. Esse esquema visa a promover equidade entre os participantes da grade, ou seja, a razão entre recursos doados e recursos consumidos deve ser próxima para todos os peers. Nesse esquema, as chances de um peer obter recursos da comunidade quando ele precisa aumentam na medida em que ele doa seus recursos ociosos para a comunidade. Para ser implantada, a rede de favores não precisa de uma infra-estrutura sofisticada tais como autoridades

certificadoras, bancos digitais ou uma moeda certificada. Logo, a rede de favores é adequada para grades computacionais peer-to-peer de escala global.

Na rede de favores, um favor é uma alocação de recurso durante um determinado tempo. Os provedores de recursos realizam favores e os consumidores recebem favores. Cada peer no sistema mantém um saldo baseado nos favores trocados com os outros peers do sistema, ou seja, para um peer A, o saldo do peer B é representado pela quantidade de favores que ele recebeu de B menos a quantidade de favores que A realizou para B. O ranking dos saldos é utilizado para priorizar o acesso aos recursos locais pelos peers da comunidade de compartilhamento de recursos. Um ataque comum realizado por participantes a sistemas baseados em ranking é sair e entrar no sistema sob uma nova identidade, este tipo de ataque é evitado pela rede de favores ao não considerar saldos negativos [22].

Quando um peer tem recursos livres, ele os doa para a comunidade. Quando dois peers remotos desejam utilizar esses recursos livres, o provedor de recursos escolhe o peer com maior saldo de favores para prover todos os recursos. Além disso, os usuários do peer local têm prioridade sobre os usuários remotos. Logo, sempre que uma aplicação de um usuário local é submetida, todas as tarefas de usuários remotos que estão executando nos recursos locais são canceladas. Isto gera badput, poder computacional que foi desperdiçado, não gerando nenhum trabalho útil.

A rede de favores é completamente descentralizada e autônoma, uma vez que não há troca das informações de saldo entre os peers, logo, não há necessidade de relações de confiança entre os peers.

Entretanto, a rede de favores assume, para seu bom funcionamento, um mecanismo de contabilidade preciso que reflita o mais fielmente possível a quantidade de recursos consumida e doada pelos peers. Se os consumidores não puderem determinar um valor preciso para os recursos consumidos, não haverá incentivo para os provedores doarem recursos de qualidade, ou seja, a melhor estratégia para os provedores será doar os piores recursos possíveis. Esse comportamento pode fazer com que o sistema entre em colapso [23], o que no OurGrid significa a inexistência de troca de favores entre os peers. Além disso, é desejável que a contabilidade também seja autônoma, para que a rede de favores possa continuar sendo implantada sem requerer nenhum acordo especial entre as partes ou acesso a serviços externos, como por

exemplo, serviços de autoridades certificadoras. Esta dissertação mostra nossa solução de um esquema de contabilidade autônoma que objetiva atender os requisitos enumerados anteriormente.

2.2 Arquitetura de Grade para Economias Computacionais (GRACE)

GRACE é um framework que apresenta modelos de economia de grade baseados em requisitos de qualidade de serviço do usuário. O objetivo é ser uma abordagem para o desenvolvimento de uma economia computacional distribuída para o compartilhamento, troca, descoberta, utilização de recursos e determinação do custo de acesso aos recursos. GRACE provê os seguintes componentes para troca de recursos: Grid Market Directory (GMD), Nimrod-G Resource Broker [10], Grid Trade Server (GTS).

2.2.1 Modelos Econômicos Propostos no GRACE

Uma forma de resolver os problemas de contabilidade e alocação de recursos em ambientes computacionais é utilizar soluções baseadas na economia real. A motivação para isso é a existência de soluções robustas já validadas, em cenários reais, para alocação de recursos. Essas soluções podem ser traduzidas para cenários computacionais. Além disso, no mundo real, mercados são naturalmente escaláveis e descentralizados. GRACE [9] descreve alguns modelos econômicos competitivos e ferramentas para auxiliar a alocação de recursos em arquitetura de computação em grade. Esses modelos são mostrados nas próximas subseções.

Modelo de Mercado de Bens

Neste modelo, os provedores especificam o preço dos serviços e contabilizam de acordo com a quantidade de recursos utilizada [9]. Os preços para o uso dos recursos mudam de acordo com a oferta e a demanda para o recurso e podem ser baseados em: taxa única, tempo de duração do uso ou em oferta e demanda. Para implementar este modelo, os provedores publicam seus preços em um serviço de páginas amarelas, o qual lista os serviços disponíveis e seus preços, para possibilitar a consulta por consumidores. Os provedores podem determinar o preço do uso de recursos com uma função de parâmetros como overhead para o serviço, importância para o usuário, preferências, etc. Esses parâmetros são de difícil obtenção, então o sistema precisa

fornecer vantagens para que os consumidores os disponibilizem [9]. Adicionalmente, os provedores precisam garantir qualidade de serviço para o período negociado.

Modelo Baseado em Negociação

No modelo apresentado anteriormente, os usuários pagam o preço estabelecido pelo provedor de recurso. A diferença é que neste modelo, o broker do usuário negocia com o provedor de recursos pelo menor preço de acesso e pelo maior período de uso. Eles negociam até que ambos alcancem um preço satisfatório. Essa negociação pode levar à baixa utilização dos recursos pelos provedores, então os provedores tendem a reduzir os preços para evitar o desperdício de ciclos no recurso. Uma das formas de negociar é o consumidor prometer favores para o provedor no futuro [9].

Modelo de Contrato e Oferta

Este modelo escolhe o provedor apropriado para realizar o trabalho de acordo com a aplicação do consumidor. Os consumidores anunciam seus requisitos e convidam os provedores a darem suas ofertas. Então, o consumidor avalia as ofertas e estabelece um contrato de uso com o provedor mais apropriado [9]. Após o uso do recurso, o provedor fornece o resultado e o valor a ser pago pelo uso baseado no contrato firmado.

Modelo de Leilão

Este modelo é particularmente útil quando existe concorrência para utilizar o recurso, ou seja, quando a demanda é maior que a oferta. Os principais papéis neste modelo são: provedores de recursos, leiloeiros e consumidores [9]. Em ambientes de grade, um protocolo de leilão auxilia provedores a decidir o preço de uso do recurso. Neste modelo, o provedor disponibiliza um anúncio do recurso e espera por ofertas dos usuários. Esse processo se repete até que o provedor receba um preço satisfatório e que os outros consumidores parem de enviar ofertas. Nesse momento o provedor oferece o recurso para o consumidor vencedor.

Modelo de Compartilhamento de Recursos Proporcional Baseado na Oferta

Este modelo aloca uma porcentagem proporcional dos recursos para os consumidores de acordo com as ofertas realizadas. Os consumidores alocam créditos para poderem acessar os recursos. O valor de cada crédito depende da demanda por recursos e é estabelecido pelo provedor. [9] afirma que este modelo é uma boa forma de distribuir os recursos de acordo com o investimento que eles desejam realizar.

2.3 Sistemas de Contabilidade de Recursos

2.3.1 GridBank

GridBank é um sistema de contabilidade e pagamento entre as partes na grade que mantém as contas dos usuários e registros de utilização dos recursos em um banco de dados [7]. O GridBank é implementado como um Web Service que pode ser acessado por provedores e consumidores de recursos. Em [7] é mostrado o protocolo de utilização do GridBank da seguinte forma (Ver Figura 1):

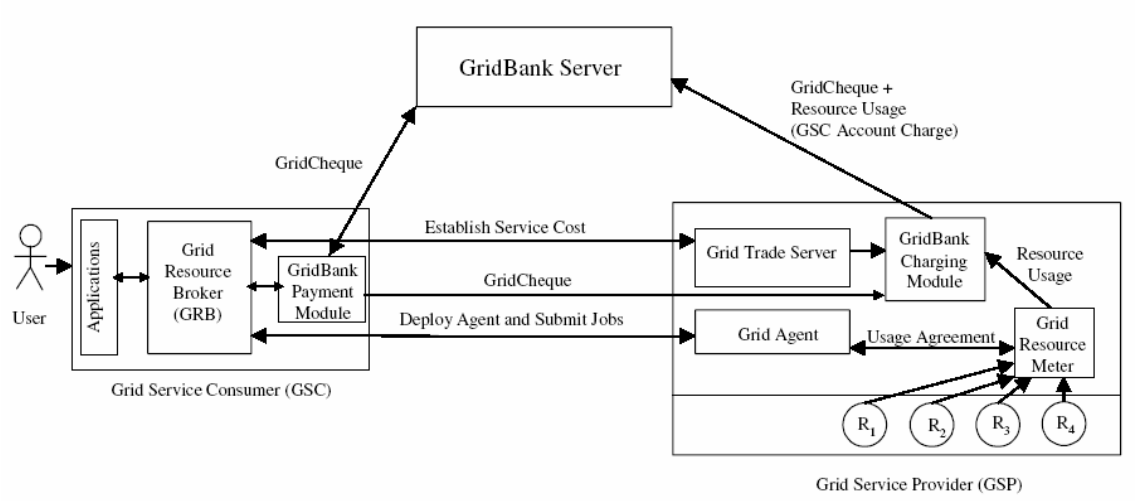

Figura 1. Protocolo de funcionamento do GridBank [7].

1. O Broker do usuário negocia o preço de utilização do recurso por unidade de tempo.
2. GridBank requisita um GridCheque do consumidor e verifica se há fundos suficientes para a utilização dos recursos.
3. O módulo de pagamento do GridBank repassa o cheque para o módulo de Charging.
4. O Broker instala o agente de grade e submete a carga de trabalho para executar no recurso.
5. O módulo de medição de utilização (GRM) pega os registros de uso de recurso de todos os recursos e repassa para o módulo GridBank Charging.
6. O módulo de Charging contacta o GridBank e registra todos os pagamentos.

Para mensurar o uso dos recursos, o GridBank tem um módulo chamado Grid Resource Meter (GRM). Esse módulo faz interface com os sistemas locais de alocação de recursos (por exemplo, escalonador) ou com os agentes de nível de usuário, que são instalados nos recursos, para extrair informação sobre o uso dos recursos. A interface com esses módulos pode ser realizada por middleware de grade como o Globus [13]. O uso é mensurado através de duas chamadas a funções nativas dos sistemas operacionais: a primeira chamada acontece após o agente de usuário terminar a execução e a outra é realizada pelo GRM para coletar os dados. Após a coleta do registro, o GRM filtra os campos relevantes e os passa para uma unidade de conversão, a qual gera um registro de uso dos recursos (RUR). O GridBank armazena esse registro no seu banco de dados e passa o RUR para o GridBank Charging Module (GBCM) para o cálculo do custo total do serviço. O GBCM obtém as taxas a serem cobradas pelo serviço do módulo Grid Trade Server (GTS) e as multiplica pelos dados contidos no RUR.

A implantação da infra-estrutura de contabilidade proposta pelo GridBank em uma grade computacional peer-to-peer de larga escala como assumimos neste trabalho não é simples, uma vez que requer que, por exemplo, um serviço de segurança como o do Globus esteja disponível para todos os participantes. Outra dificuldade para a adoção do GridBank no modelo assumido por nós é que o uso dos recursos é medido apenas no lado do provedor pelo GRM. Como assumimos que não há confiança entre as partes envolvidas, o provedor de recursos poderia informar um valor errado para o servidor GridBank e com isso ganhar créditos maiores pela execução das aplicações.

2.3.2 Sistema de Contabilidade do SweGrid (SGAS)

O SweGrid [21] é uma grade computacional composta de seis clusters computacionais, com cerca de 600 CPUs, conectados através de uma rede de alto-desempenho. O SweGrid fornece tempo de computação, medido em horas por processador, para projetos de pesquisa. O SGAS [20] é um sistema de contabilidade de grade que mantém uma visão única de toda a grade sobre os recursos consumidos pelos membros de uma organização virtual (VO). Cada VO mantém um serviço de banco, que é o serviço principal do sistema de contabilidade responsável por receber pagamento e mensurar o valor de uso dos recursos. Uma hipótese assumida pelo SGAS é que há um relacionamento de confiança pré-existente entre as entidades participantes da comunidade.

O componente de banco do SGAS foi desenvolvido para contabilizar qualquer tipo de recurso e se integrar a qualquer ambiente de Grade. As transações são realizadas utilizando créditos de grade, uma moeda abstrata que é utilizada para contabilizar as utilizações de recursos. A contabilidade é realizada transformando os diferentes tipos de medida de uso do recurso em créditos de Grade.

Os recursos disponíveis atualmente no Swegrid são homogêneos, isto é, todas as máquinas de todos os sites têm configuração igual. A única métrica de contabilidade é o tempo de disponibilidade do recurso. No caso do Swegrid, o mapeamento de uso de recurso para créditos de grade é trivial. [20] argumenta que é necessário uma evolução do sistema capaz de lidar com diferentes tipos de recursos, por exemplo, recursos mais rápidos, que deveriam ser mais caros. Uma estratégia sugerida é que sejam fornecidas contas diferentes para cada tipo de recurso, sendo diferente o valor da conversão de uso de recursos para créditos de grade.

Apesar da contabilidade no SGAS ser distribuída por vários componentes de banco, eles interagem para manter um registro consistente dos créditos da grade disponível. Durante esta interação, os provedores poderiam informar um valor errado para a contabilidade se não fosse assumido um relacionamento de confiança entre as partes. Além disso, a contabilidade é facilitada porque todos os recursos da grade são iguais, o que torna possível a utilização apenas do tempo como métrica de contabilidade. Porém, como mostramos no Capítulo 4 e no Capítulo 5, em um cenário mais heterogêneo, contabilizar utilizando apenas o tempo pode incentivar os provedores a disponibilizarem recursos lentos na grade.

2.3.3 Sistema de Contabilidade Distribuída

[15] argumenta que para contabilizar recursos em ambientes distribuídos, uma única métrica, unidades de alocação de grade, para representar o uso de recursos, precisa ser criada. Essas unidades de alocação precisam ser distribuídas entre os provedores de acordo com a quantidade de recursos. O valor dos recursos é dado através de fatores de peso obtidos através da curva de oferta e demanda para o recurso. A utilização de curvas de oferta e demanda ajuda a criar um mercado de recursos que balanceia a carga de uso dos recursos. Os usuários que não possuem recursos para doar à grade precisam se associar com um domínio administrativo que tenha recursos. Nesse caso, é necessária uma forma de transferir fundos entre domínios administrativos.

Uma dificuldade na implantação desse modelo é a necessidade de estimar o valor dos recursos antes de executar as aplicações. Além disso, ele requer uma infra-estrutura que suporte contabilidade distribuída que consiga prover as funcionalidades de troca de informações de contabilidade, alocação de recurso e gerenciamento de cotas de uso e informações de contabilidade entre consumidores e provedores [14].

2.3.4 Sistema de Contabilidade do DataGrid (DGAS)

O sistema de contabilidade DGAS é um protótipo de um sistema de contabilidade desenvolvido para o DataGrid [26]. O DGAS é composto de um conjunto de servidores geograficamente distribuídos, chamados de Home Location Register (HLR), que pode ser visto como um serviço de banco que armazena contas para consumidores e provedores de recursos [25]. No DGAS, os administradores de sites recebem créditos ao fornecerem recursos para a execução de aplicações de usuários da grade. A execução de uma aplicação deve ser autorizada antes de sua submissão para a grade baseada no número de créditos que o consumidor tem na grade. Para a autorização do uso da grade, deve ser fornecida uma estimativa da quantidade de recursos computacionais que uma aplicação precisará antes de sua execução. [27] argumenta que essa estimativa pode ser fornecida pelo próprio usuário, na descrição da aplicação, ou pode ser calculada baseada em históricos de execuções anteriores da mesma aplicação. Essa estimativa é multiplicada pelo preço de uso do recurso obtido dos HLRs, para verificar se o usuário tem crédito suficiente para executar a aplicação.

O DGAS assume que existe um monitor de uso instalado nos recursos e que esse monitor fornece o registro de utilização dos recursos por uma aplicação para os HLRs. A contabilidade do uso dos recursos é realizada multiplicando o valor fornecido pelo monitor pelo preço fornecido pelo HLRs no momento da submissão da aplicação. Após a execução, o HLR transfere os créditos do consumidor para o provedor de recursos.

Para verificar se o uso dos recursos foi corretamente medido pelo provedor, o consumidor poderia utilizar a estimativa realizada antes da execução da aplicação. Porém, nem sempre a estimativa para execução de uma aplicação está disponível ou é confiável o suficiente para confrontá-la com o uso de recursos mensurado pelo monitor. Isso torna possível que o provedor forneça resultados errados para o uso do recurso, sem necessariamente ser detectado.

2.3.5 Contabilidade baseada em BenchMark

Uma característica importante de um sistema de contabilidade é a necessidade de uma unidade de contabilidade comparável. Isso é importante para que consumidores e provedores possam analisar se a contabilidade do uso do recurso é coerente e alimentar os sistemas econômicos ou de reputação de maneira correta.

A primeira estratégia pensada por nós para realizar contabilidade foi fazer com que os consumidores enviassem micro-benchmarks para os provedores. Esses micro-benchmarks são pequenos programas cujo tempo de execução é conhecido em uma máquina de referência. O tempo em que um destes pequenos programas leva para executar no recurso de um provedor pode avaliar o estado corrente do recurso e pode ser usado para estimar o poder computacional real do recurso.

O funcionamento do esquema de contabilidade é simples; as tarefas são submetidas para provedores junto com tarefas de micro-benchmark para avaliar o poder computacional corrente do recurso alocado pelo provedor para executar a tarefa. O tempo de execução das tarefas de benchmark é retornado para o consumidor, que usa este tempo para normalizar o valor de cada disponibilização de recurso. A contabilidade *Acc* da disponibilização de recurso *f* é realizada da seguinte forma:

$$Acc(f) = t(f) / t(b)$$

Em que *t(f)* é o tempo de duração da utilização do recurso e *t(b)* é o tempo gasto para executar a tarefa de benchmark.

Um problema nessa solução é que provedores maliciosos podem identificar as tarefas de benchmark e fraudá-las. O provedor pode retornar um valor antes de ter executado a tarefa completamente. A conseqüência disso é que o consumidor acreditaria que o poder computacional do recurso é maior do que ele realmente é.

Outro problema desse esquema é lidar com arquiteturas computacionais diferentes visto que um tipo de benckmark pode ser mais adequado para uma delas. Além disso, o consumidor não tem nenhuma garantia de que a tarefa será executada no mesmo recurso no qual a tarefa de benckmark foi executada. Desta forma, um possível ataque à contabilidade seria o provedor executar as tarefas de *benckmark* em recursos rápidos e as tarefas das aplicações de usuários em recursos mais lentos.

CAPÍTULO 3

Metodologia

Este capítulo descreve a metodologia que utilizamos para avaliar os esquemas de contabilidade autônoma. Primeiro descrevemos o modelo de grade utilizado. Esse modelo foi baseado no OurGrid e representa o escalonamento de aplicações em peers e as doações de recursos entre domínios administrativos diferentes. Após isso, mostramos o simulador que utilizamos para implementar o modelo de grade e também em que cenários avaliamos os esquemas de contabilidade. Em seguida, apresentamos a forma como definimos o intervalo de confiança de nossas simulações, bem como a plataforma que utilizamos para executá-las. Finalmente, mostramos a forma que utilizamos para analisar os esquemas de contabilidade e as métricas que utilizamos.

3.1 Modelo de Grade

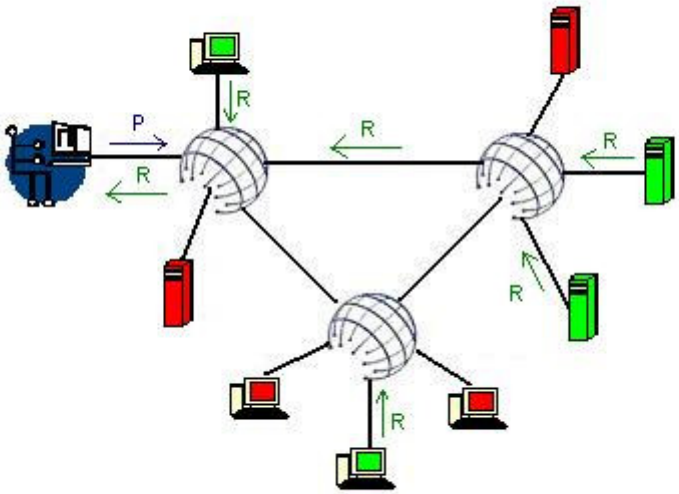

Figura 2. Requisição de recursos de um consumidor a comunidade no modelo de grade peer-to-peer.

No modelo utilizado neste estudo (Ver Figura 2), a grade é composta por um conjunto de peers, os quais, por sua vez, possuem um conjunto de recursos ou máquinas. Cada máquina tem um poder computacional - capacidade que a máquina tem de processar unidades de trabalho em uma unidade de tempo. O poder computacional

do peer como um todo é a soma dos poderes de todos os seus recursos. O número de máquinas por peer e o poder dessas máquinas podem variar.

Os peers se unem formando uma comunidade de compartilhamento de recursos. Nessa comunidade, cada peer pode atuar como consumidor e como provedor. O peer atua como consumidor quando um cliente submete aplicações diretamente para ele. Nesse caso, o peer disponibiliza todos os recursos locais livres ou alocados para outros peers para executar as tarefas da aplicação e faz uma requisição de recursos para os peers da comunidade. Quando uma requisição de um cliente local chega, as tarefas das requisições oriundas dos outros peers da comunidade são canceladas e os recursos necessários são destinados às requisições locais. Quando uma tarefa é cancelada, uma réplica dessa tarefa é criada pelo consumidor e adicionada à requisição original, isso garante que todas as tarefas das aplicações serão executadas. O poder computacional gasto com as tarefas canceladas é considerado badput. Quando um peer atende à requisição de outro peer, ele atua como provedor. Note que, para incentivar os provedores a disponibilizarem recursos, apenas os recursos ociosos são doados e os peers têm prioridade na execução de suas aplicações em seus próprios recursos.

Cada requisição representa uma submissão de uma aplicação feita ao peer e cada aplicação é composta por um conjunto de tarefas independentes, ou seja, as tarefas não precisam de comunicação durante a execução. Cada tarefa tem um custo computacional. O número de máquinas requisitadas para o atendimento da requisição é igual ao número de tarefas da aplicação.

Quando um peer está provendo recursos para a comunidade, ele usa o esquema de rede de favores para definir a alocação de recursos [1]. Na rede de favores, todas as alocações de recursos são consideradas favores e um ranking é construído baseado no balanço de troca de favores com os demais peers da comunidade. Para alimentar esse ranking, existe um componente responsável por contabilizar os favores recebidos e fornecidos, que é o alvo desta dissertação.

Em uma grade computacional peer-to-peer, um dos fatores que podem afetar as contribuições no sistema é a sabotagem, retorno de resultados errados pelo provedor para as tarefas do consumidor [11]. Um dos objetivos dos provedores é obter retorno econômico dos consumidores sem utilizar os seus recursos. O resultado desse comportamento pode influenciar a contabilidade de recursos e, pior ainda, comprometer

parte da computação realizada [17]. Em nosso trabalho, assumimos que todas as tarefas contabilizadas nos favores estão livres de problemas de sabotagem, ou seja, existe um sistema de detecção de sabotagem como o descrito em [4].

3.2 Simulador

Como forma de avaliar os esquemas de contabilidade em uma gama de cenários distintos, nós desenvolvemos um simulador para o nosso modelo. Nesse simulador, alguns parâmetros que podem ser variados para a execução das simulações são:

- *Número de peers da comunidade*: Nós variamos a quantidade de peers da comunidade para verificar como a contabilidade afeta a cooperação em comunidades de tamanhos diferentes. O número de peers nas simulações variou de 10 a 100. Como o tamanho das comunidades não influenciou os resultados alcançados, nos detemos em comunidades entre 15 e 20 peers, que é o tamanho médio que tem hoje a comunidade *OurGrid*, uma comunidade aberta para compartilhamento de recursos baseada em tecnologia peer-to-peer [19].

- *Poder médio das máquinas da grade*: A variação do poder médio das máquinas foi realizada para verificar se o esquema de contabilidade era justo para peers com máquinas diferentes. A retribuição de favores proporcional à potência das máquinas é importante para que os administradores de domínios sejam incentivados a disponibilizarem máquinas rápidas para a comunidade.

- *Número de tarefas de cada aplicação*: Esse parâmetro permite verificar a adequação dos esquemas de contabilidade a aplicações de tamanho diferentes.

- *Tamanho das tarefas*: Nosso objetivo é verificar a adequação dos esquemas de contabilidade a aplicações homogêneas, aplicações nas quais há pouca diferença no custo computacional das tarefas - como algumas aplicações de processamento de imagem - e heterogêneas, aplicações que as tarefas apresentam grande diferença de custo computacional, como aplicações que procuram chaves em uma quantidade de dados.

- *Intervalo entre submissões de aplicação*: Esse parâmetro é calculado de forma a saturar, em média, os peers da comunidade. O objetivo é garantir que os peers estejam consumindo e doando recursos durante o tempo simulado. A saturação dos peers nos permite observar perturbações causadas por provedores de máquinas lentas

ou free-riders na comunidade, o que não seria possível se a comunidade tivesse, em média, mais recursos disponíveis do que a capacidade de consumi-los. O cálculo do intervalo médio entre submissões de aplicações é realizado dividindo a carga de trabalho pelo poder do peer.

- *Número de aplicações submetidas por peer*: Utilizamos esse parâmetro para variar o tempo simulado, ou seja, aumentamos o número de aplicações para aumentar o tempo simulado.

A alteração no número e no poder médio das máquinas nos permitiu alterar o poder do peer, ou seja, a soma dos poderes de todos os recursos de um peer. Esse poder representa quantas unidades de trabalho um peer consegue processar em uma unidade tempo. A variação na potência média entre peers nos permitiu montar comunidades com peers de composição interna diferente, variando desde peers com máquinas lentas a peers com máquinas mais rápidas.

As alterações nas aplicações permitiram avaliar a influência da carga submetida ao peer para os esquemas de contabilidade. As aplicações podem ser alteradas devido a fatores como diminuição ou aumento do custo das tarefas da aplicação e alterações no número de tarefas.

Para avaliar os esquemas de contabilidade, nós variamos os parâmetros discutidos acima constituindo cenários de simulação. Para a construção desses cenários, desenvolvemos um gerador de cenários que é capaz de construir uma instância de cenário a partir de um arquivo de configuração XML (eXtended Markup Language) (Ver Figura 3). Separamos o gerador de cenários do restante do simulador porque precisamos executar cada cenário de simulação para os três esquemas de contabilidade (perfeita, por tempo e por poder relativo).

Em [2][3], Casanova et al. realizaram um estudo sobre as características dos ambientes de grade e mostraram que a distribuição das velocidades das máquinas na grade obedece a distribuições uniformes. A observação da comunidade OurGrid, utilizada para a execução de nossas simulações, também contribuiu para a elaboração desse cenário, uma vez que verificamos a existência de peers com máquinas de velocidades médias diferentes.

```
- <scenario>
    <accountingScheme>1</accountingScheme>
    <numPeers>10</numPeers>
    <minNumGms>25</minNumGms>
    <maxNumGms>25</maxNumGms>
    <minGmPower>4</minGmPower>
    <maxGmPower>16</maxGmPower>
    <numJobsPerPeer>1000</numJobsPerPeer>
    <minNumTask>250</minNumTask>
    <maxNumTask>250</maxNumTask>
    <minInterArravialTime>1</minInterArravialTime>
    <maxInterArravialTime>799</maxInterArravialTime>
    <minTaskSize>400</minTaskSize>
    <maxTaskSize>400</maxTaskSize>
    <minPeerPower>250</minPeerPower>
    <maxPeerPower>250</maxPeerPower>
  </scenario>
```

Figura 3. Exemplo de arquivo de descrição de cenários.

Os cenários são gerados utilizando distribuições uniformes baseadas nos valores mínimos e máximos existentes no arquivo de configuração. Por exemplo, o número de máquinas de um peer é calculado utilizando uma distribuição uniforme com valores máximos e mínimos, respectivamente *maxNumGms* e *minNumGms*[1]. Logo, se os valores mínimo e máximo do parâmetro forem iguais, o parâmetro será homogêneo nesse cenário.

O resultado do programa gerador de cenários é uma instância de cenário descrita em três arquivos XML que descrevem os peers, as aplicações e as submissões.

No arquivo que descreve os peers são definidas as identificações e todas as instâncias das máquinas de cada peer determinando a potência que cada máquina terá.

No arquivo que descreve as aplicações são definidas a identificação e a quantidade de tarefas da aplicação. Esse arquivo descreve também todas as tarefas que compõem a aplicação e o custo computacional delas.

O arquivo de submissões associa as aplicações aos peers, ou seja, cada aplicação é associada ao peer no qual ela será submetida. O tempo simulado em que cada aplicação será submetida também é descrito nesse arquivo.

[1] Gm ou GuM é um acrônimo para *Grid Machine*

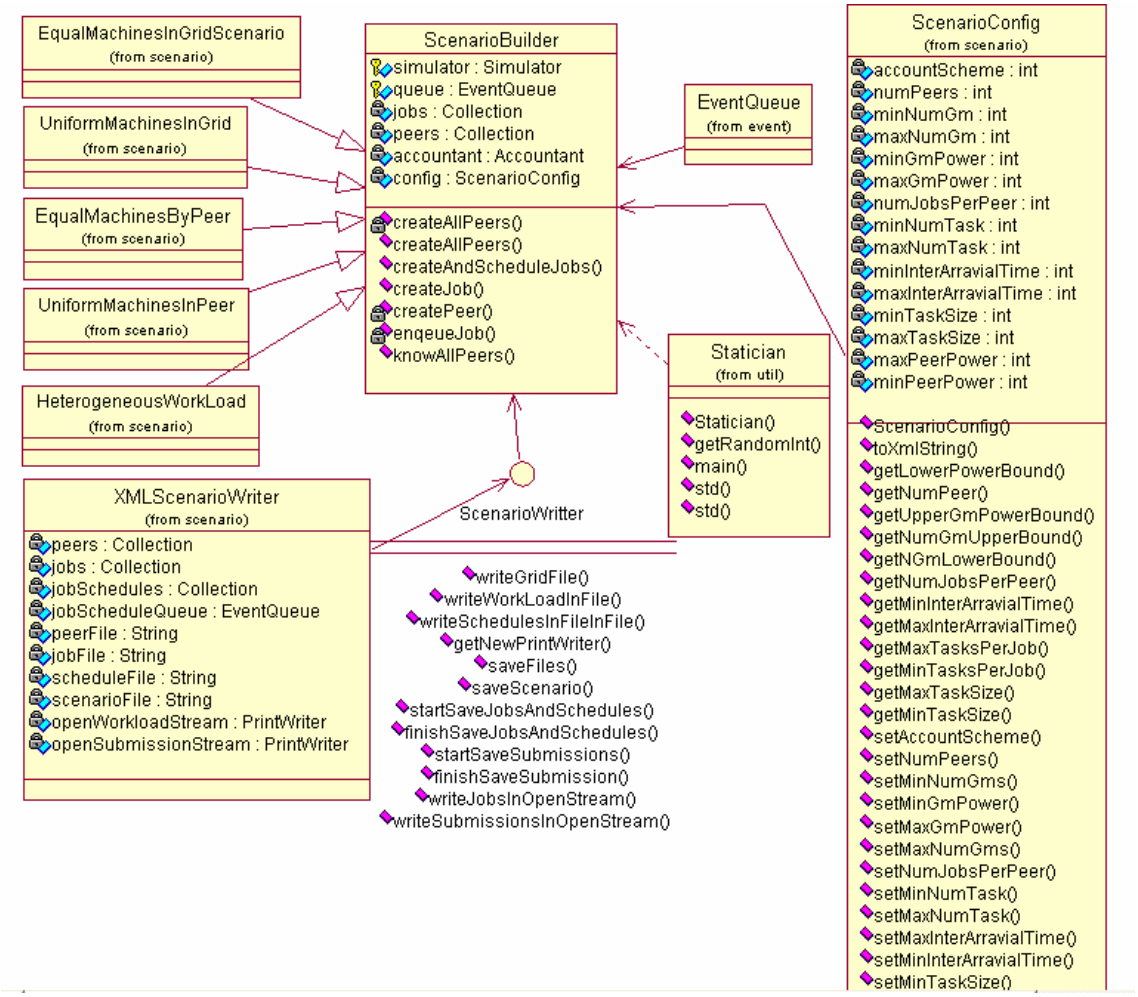

Figura 4. Núcleo do diagrama de classes do gerador de cenários.

A Figura 4 mostra a parte principal do diagrama de classes do gerador de cenários construído. A classe *ScenarioConfig* é a classe responsável por ler uma especificação de cenário como, por exemplo, a da Figura 3 e mantê-la em memória. A classe *ScenarioBuilder* é a classe responsável por construir os cenários de simulação a partir de uma configuração. Para isso, todos os peers, aplicações (jobs) e submissões são criados. Os peers são escritos em um arquivo XML que descreve os peers e suas máquinas. Em alguns cenários, as máquinas têm potências iguais e, em outros, as potências das máquinas são escolhidas de acordo com uma distribuição uniforme. A aleatoriedade das amostras da distribuição uniforme é garantida pela classe *Statician*. As tarefas das aplicações são criadas seguindo a aleatoriedade da distribuição uniforme definida nos parâmetros de configuração. Depois de criadas, as aplicações são associadas aos peers aos quais serão submetidas. Os tempos mínimo e máximo entre submissões de aplicações, lidos do arquivo, são utilizados para escolher o momento em que as aplicações serão submetidas. As submissões de todos os peers são convertidas

em eventos e ordenadas pelo tempo simulado pela classe *EventQueue*. A classe *ScenarioWriter* é a classe responsável por escrever os peers, as aplicações e as submissões em arquivos XML.

Para avaliar os esquemas de contabilidade autônoma foi desenvolvido um simulador baseado em eventos para grades computacionais peer-to-peer. Nesse simulador são descritas as entidades necessárias para a simulação da computação em grade como peers, aplicações, máquinas e tarefas.

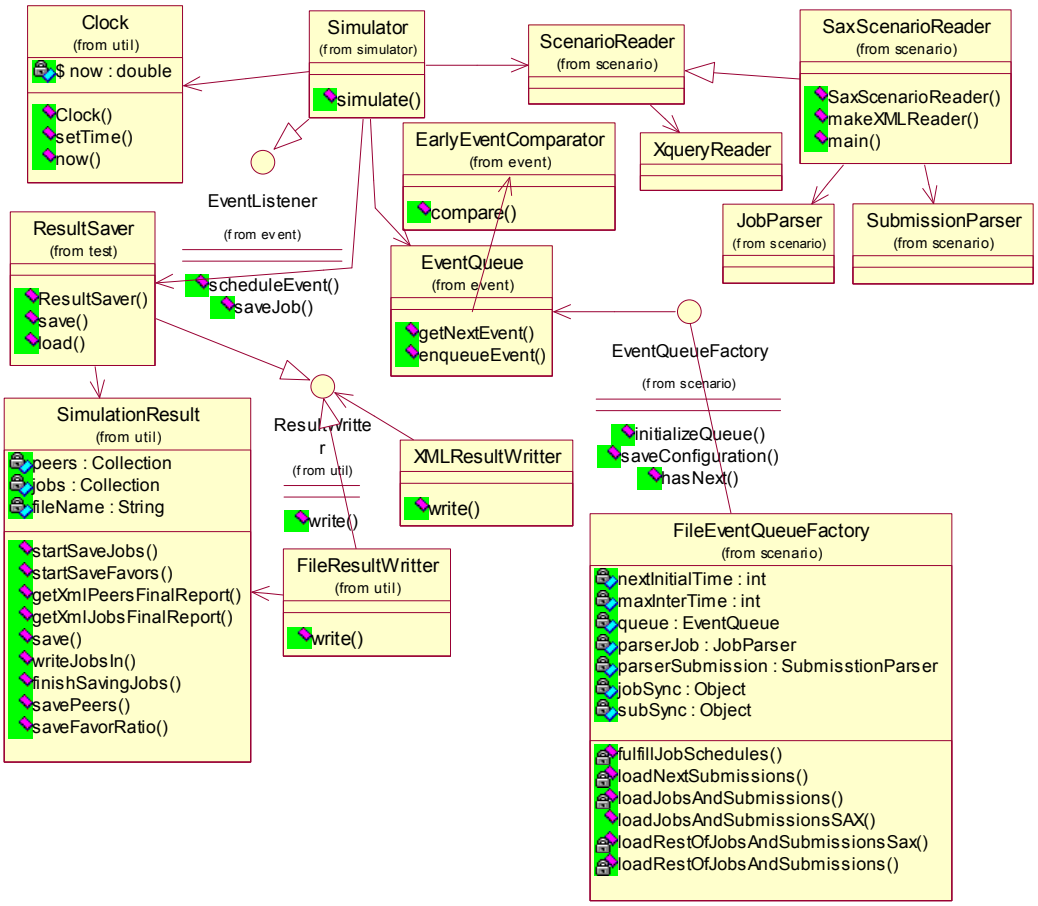

Figura 5. Diagrama da leitura de cenários, processamento e escrita de resultados.

A Figura 5 ilustra as classes responsáveis pela leitura dos cenários, processamento dos eventos e escrita dos resultados. A classe Simulator utiliza o *ScenarioReader* para ler os cenários produzidos pelo gerador de cenários. A leitura utiliza Sax, uma biblioteca de parser seqüencial XML, para ler os arquivos XML. Os peers são lidos e instanciados em memória no início da simulação. Na implementação inicial, a quantidade de eventos e de aplicações (jobs) simulada foi suficiente para causar problemas de falta de memória em uma máquina com 512 megabytes de

memória RAM. Para solucionar esse problema, os eventos foram carregados sob demanda e foram descartados logo após serem processados.

As ações realizadas pelas entidades do simulador são modeladas como eventos (Ver Figura 6). Os eventos são criados, por exemplo, quando tarefas terminam ou aplicações são iniciadas. Quando o evento é lido na fila, é chamado um método para processá-lo, em que o evento manda as mensagens necessárias para as entidades que devem ser avisadas da sua ocorrência. Note que o processamento de um evento pode produzir novos eventos a ser enfileirados descrevendo o comportamento de uma grade peer-to-peer que utiliza a rede de favores para priorizar requisições (Ver Figura 7).

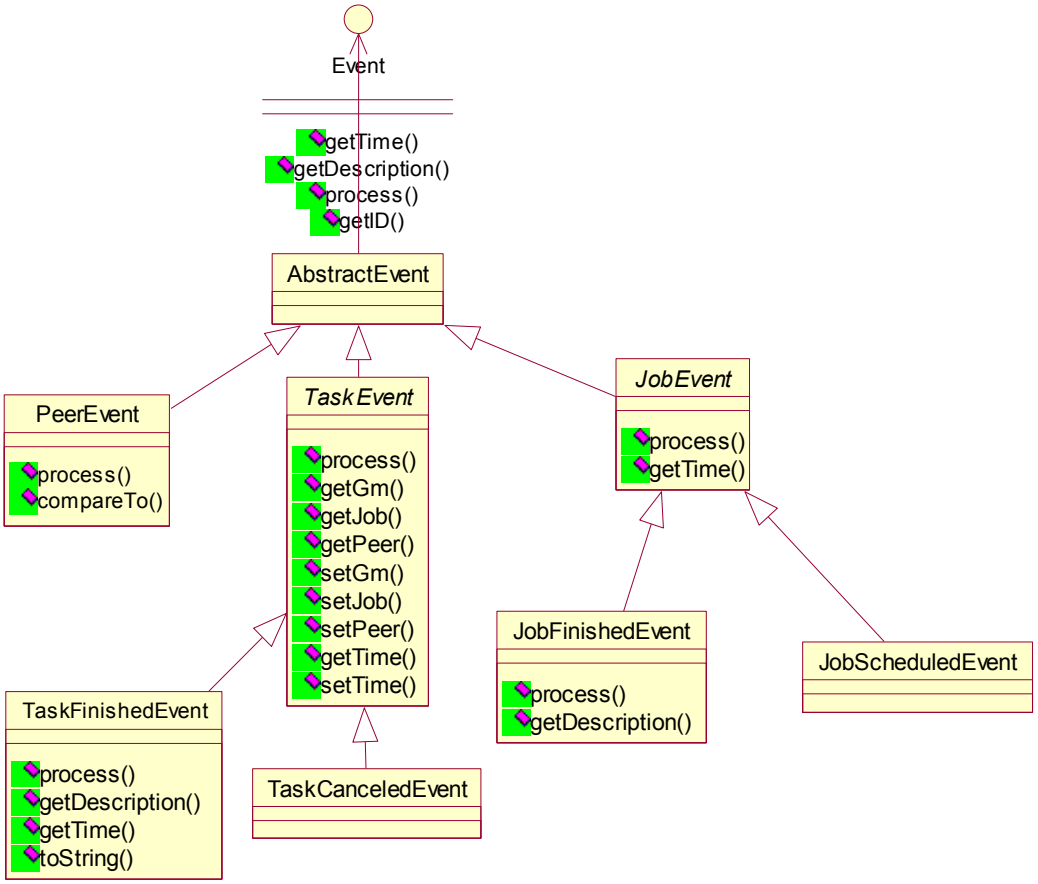

Figura 6. Eventos do simulador.

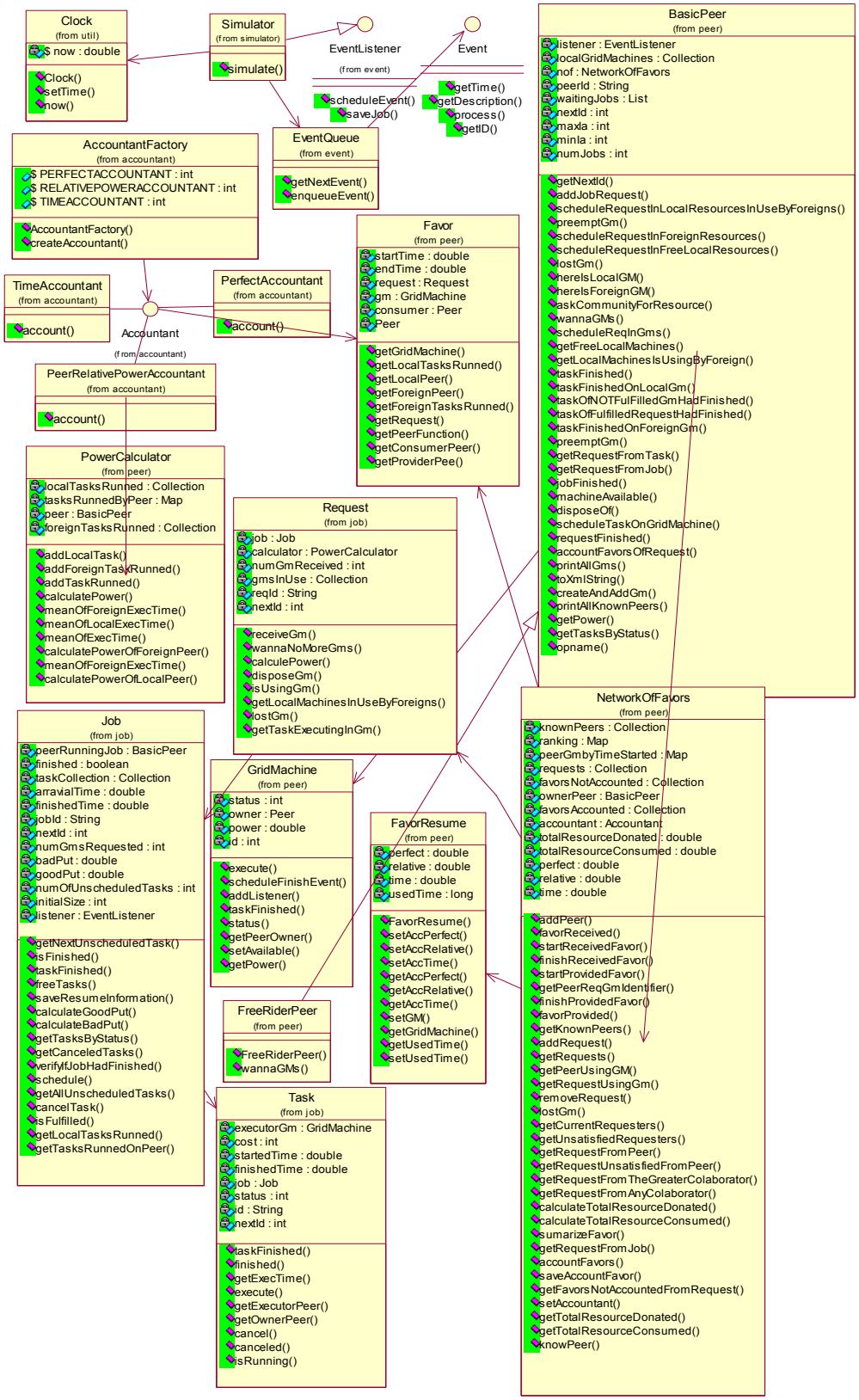

Figura 7: Diagrama de classes do núcleo do simulador.

A simulação é finalizada quando todos os eventos da fila de eventos são processados. Nesse momento são gerados arquivos XML descrevendo o resultado da simulação. Esses arquivos descrevem como as aplicações, os peers e os favores terminaram. Por exemplo, no arquivo de resultado de aplicações, há informações do tempo simulado em que a aplicação foi submetida e no qual ela terminou, além de informações sobre o badput (processamento que não gerou nenhum resultado útil). Já no resultado de peers, são armazenadas informações a respeito da quantidade de recursos fornecidos e recebidos durante a simulação.

3.3 Definição de Cenários de Simulação

Estudamos cenários de simulação que representassem mudanças nas máquinas dos peers e nas tarefas das aplicações. A variação partiu de cenários totalmente homogêneos para cenários nos quais os peers tivessem máquinas diferentes e as aplicações tivessem tarefas de diferentes custos computacionais. O objetivo dessa variação é verificar a adequação de cada esquema de contabilidade a cada cenário e com isso identificar os pontos fortes e fracos de cada um.

Em todos os cenários, variamos a carga de trabalho proporcionalmente ao poder de cada peer, ou seja, quanto maior o poder do peer, mais trabalho ele recebe. O objetivo dessa variação foi simular cenários em que havia contenção de recursos. Embora os esquemas de contabilidade funcionem independentemente da carga submetida aos peers, a avaliação dos esquemas de contabilidade em uma situação em que haja contenção de recursos é necessária, visto que se a carga de trabalho fosse menor que a oferta de recursos todos os peers conseguiriam os recursos que quisessem.

[1] mostrou que, em contenção, utilizando o esquema de rede de favores, a quantidade de recursos que os peers recebiam da comunidade era proporcional a de suas doações. Assim, variamos o tamanho da aplicação para que, em média, todos os peers estivessem saturados, ou seja, fizemos o intervalo médio entre chegadas de aplicações *ia* ser a divisão da carga de trabalho pelo poder do peer seguindo a relação:

$$ia = \frac{nT * tT}{nGM * pGM}$$

Em que *nT* é a quantidade média de tarefas, *tT* é o tamanho médio dessas tarefas, *nGM* é o número de máquinas do peer e *pGM* é o poder médio dessas máquinas. Como

o intervalo entre chegadas de aplicações varia em torno da média, obtivemos momentos em que cada peer tinha recursos disponíveis para realizar doações para a comunidade e, também, momentos nos quais eles tinham uma carga de trabalho maior do que sua capacidade de processamento e necessitavam de recursos da comunidade para terminar a execução da aplicação em menos tempo. A seguir apresentamos os cenários nos quais os esquemas de contabilidade foram avaliados.

Cenário 1

O primeiro cenário investigado é o caso onde as aplicações e os peers são totalmente homogêneos, ou seja, não há variação no tamanho ou na quantidade de tarefas de cada aplicação nem no número ou potência das máquinas de cada peer.

Em cada execução foram instanciados 15 peers com 25 máquinas de poder 10, assim, o poder dos peers é igual a 250. O custo computacional das aplicações foi mantido constante em 100000 unidades de trabalho. Cada aplicação contém 250 tarefas com custo computacional igual a 400. O intervalo entre a chegada de aplicações para cada peer foi variado em uma distribuição uniforme de 1 a 799 para saturar os peers. Note que isso satura a grade, o que é importante para podermos avaliar a influência dos esquemas de contabilidade no sistema em cenários em que haja competição por recursos.

O objetivo desse cenário é verificar se o simulador desenvolvido neste trabalho apresenta resultados similares ao alcançado em [1] . Em [1], foi mostrado que utilizando o esquema de rede de favores para incentivar o compartilhamento de recursos em grades peer-to-peer, a razão de favores recebidos sobre favores realizados dos peers tendia a 1, assumindo a existência de um esquema de contabilidade perfeita que seria capaz de alimentar a rede de favores.

Cenário 2

O objetivo do cenário 2 é verificar como os esquemas de contabilidade se comportam quando as tarefas que compõem cada aplicação têm custos computacionais diferentes.

Esse cenário apresenta a mesma configuração da grade do cenário passado, porém, as aplicações são heterogêneas, ou seja, têm tarefas de custos computacionais diferentes variando conforme distribuição uniforme U(100,700). Note que, a média do custo computacional das tarefas continua igual a 400 e a grade continua saturada.

Cenário 3

O objetivo do cenário 3 é investigar as conseqüências de mudanças internas nos recursos do peer em relação à contabilidade dos recursos. Nesse caso, as máquinas de cada peer não têm necessariamente o mesmo poder computacional, porém a média dos poderes de todas as máquinas de cada peer é igual a 10.

Nesse sentido, concebemos um cenário caracterizado pela homogeneidade estatística dentro dos peers e entre peers. Em cada simulação, foram instanciados 15 peers com 250 unidades de poder computacional cada. O poder computacional de cada peer foi dividido em 25 máquinas de poder médio 10. Embora a média do poder das máquinas de todos os peers tenha se mantido constante em 10, o poder das máquinas de todos os peers foi variado seguindo uma distribuição uniforme de 4 a 16. Foram submetidas 1000 aplicações para cada peer, cada uma com 250 tarefas de 400 unidades de trabalho e com intervalo entre a chegada das aplicações variado seguindo a distribuição uniforme U(1,799).

Cenário 4

O cenário 4 avalia o comportamento dos esquemas de contabilidade quando os peers tentam obter vantagens do sistema doando recursos de baixa qualidade. Em uma grade peer-to-peer é importante que os consumidores consigam identificar os recursos de baixa qualidade e atribuir um valor menor para o uso destes recursos do que para os recursos rápidos. Se os provedores receberem o mesmo crédito para doações não importando o quanto seus recursos são lentos, eles são incentivados a rodar várias tarefas em uma mesma máquina, "transformando" uma máquina rápida em várias lentas e o sistema entra em colapso [23].

Nesse cenário, a grade tem 20 peers, cada um com 25 máquinas de poder computacional variando conforme a distribuição uniforme U(4,16). Entretanto, os peers estão divididos igualmente em duas classes: peers lentos e peers rápidos. Os peers lentos executam duas tarefas simultaneamente em cada máquina, simulando para a grade ter 50 máquinas de poder U(2,8). Já os peers rápidos alocam uma tarefa por vez para cada máquina. As aplicações chegam a cada U(1,799) unidades de tempo, cada uma com 250 tarefas de custo computacional U(200,600).

Cenário 5

O objetivo do cenário 5 é verificar o comportamento dos esquemas de contabilidade em comunidades nas quais os peers têm máquinas de poder computacional diferente, ou seja, alguns peers têm recursos rápidos, enquanto outros têm recursos lentos. Esperamos com isso verificar se os esquemas de contabilidade incentivam os provedores de recursos a disponibilizarem recursos rápidos na grade.

Para isso, definimos três tipos de peers; (i) peers com máquinas de poder computacional igual a 4, (ii) peers com máquinas de poder 10 e (iii) peers com máquinas de poder 16.

Nesse cenário, mantivemos o número de máquinas por peer constante e o poder dos peers varia de acordo com a potência de suas máquinas. Para saturar a grade e gerar competição por recursos, aumentamos o número de tarefas de cada aplicação proporcionalmente ao poder total do peer. Desta forma, o número de aplicações foi igual a 100, 250 e 400 para os peers com máquinas de poder médio igual a 4, 10 e 16 respectivamente. Cada aplicação submetida tinha custo computacional suficiente para ser processada 400 unidades de tempo, que é o tempo médio entre a chegada de aplicações.

Cenário 6

O objetivo do cenário 6 é avaliar os esquemas de contabilidade no caso em que os peers tenham potência média diferente e as máquinas dos peers tenham potência diferente, ou seja, diferentemente do cenário anterior, não existe apenas um tipo de máquina por peer.

Nesse cenário, mantivemos o poder de todos os peers igual a 250 e consideramos três classes de peer. A distribuição dos poderes das máquinas das três classes de peer foi: U(2,6), U(5,15), U(8,24). Nesse caso, os peers seguem com máquinas de poder médio em 4, 10 e 16, porém há uma variação da máquina mais lenta para a máquina mais rápida. Note que esse cenário é mais heterogêneo que os anteriores e os peers não necessariamente têm o mesmo número de máquinas, nem mesmo para os peers da mesma classe. As aplicações desse cenário são iguais às do cenário 5 e têm 250 tarefas com custo computacional igual a 400.

Cenário 7

O cenário 7 avalia o caso em que a grade é formada por peers com recursos de poder computacional heterogêneo e o custo computacional das tarefas que compõem as aplicações variam. O objetivo é avaliar os esquemas de contabilidade no cenário que esperamos ser mais freqüente na prática.

Nesse cenário, o poder dos peers varia de acordo com a potência de suas máquinas. As máquinas dos peers não são equivalentes. Definimos três tipos de peers: (i) peers com máquinas de poder médio 4 (variando de 2 a 6), (ii) peers com máquinas de poder médio 10 (variando de 5 a 15) e (iii) peers com máquinas de poder 16 (variando de 8 a 24). A carga de trabalho é proporcional à potência dos peers. Cada aplicação tem 250 tarefas com custo computacional variando de acordo com uma distribuição uniforme U(200,600). A Tabela 1 e a Tabela 2 mostram resumos dos principais atributos da grade e das aplicações nos cenários estudados:

	Número de peers por simulação	**Número de máquinas por peer**	**Poder computacional de cada peer**	**Distribuição do poder computacional das máquinas**	**Heterogeneidade dos recursos entre peers**
Cenário 1	15	25	250	U(10,10)	Todos os peers com máquinas de poder 10.
Cenário 2	15	25	250	U(10,10)	Todos os peers com máquinas de poder 10.
Cenário 3	15	25	250	U(4,16)	Máquinas de todos os peers seguem a mesma distribuição.
Cenário 4	20	10 simulando ter 50 máquinas (peers lentos) 10 com 25 máquinas (peers rápidos)	250	Peers lentos – U(2,8) Peers Rápidos – U(4,16)	Cada peer tem máquinas de uma classe.
Cenário 5	15	25	Varia de acordo com a potência das máquinas	Três classes de peers: U(4,4) U(10,10) U(16,16)	Cada peer tem máquinas de uma classe.

Cenário 6	15	Em média: 83 25 17	250	Três classes de peers: U(2,6) U(5,15) U(8,24)	Cada peer tem máquinas de uma classe.
Cenário 7	15	Em média: 83 25 17	250	Três classes de peers: U(2,6) U(5,15) U(8,24)	Cada peer tem máquinas de uma classe.

Tabela 1. Resumo da grade nos cenários avaliados.

	Número de tarefas	**Tamanho das tarefas**	**Tempo entre chegada de aplicações por peer**	**Número de aplicações submetidas por peer**	**Custo computacional das aplicações**
Cenário 1	250	400	U(1,799)	1000	100000
Cenário 2	250	U (100,700)	U(1,799)	1000	100000, em média
Cenário 3	250	400	U(1,799)	1000	100000
Cenário 4	250	U (200,600)	U(1,799)	1000	100000
Cenário 5	Varia de Acordo com o poder do peer (igual ao poder do peer) 100, 250, 400	400	U(1,799)	1000	40000 100000 160000
Cenário 6	250	400	U(1,799)	1000	100000
Cenário 7	250	U (200,600)	U(1,799)	1000	100000, em média

Tabela 2. Resumo das aplicações nos cenários avaliados.

3.4 Intervalo de Confiança

Para assegurar que as amostras analisadas representavam de maneira correta os cenários, nós definimos o nível de confiança que as amostras deveriam atingir para que pudessem ser analisadas. O nível de confiança escolhido foi de 95% por nos permitir uma confiabilidade satisfatória dos resultados em um número de execuções factíveis.

Executamos um número reduzido de simulações para auxiliar no cálculo do número de amostras necessário para atingir o intervalo de confiança. Para calcular o número de amostras *N* utilizamos a seguinte fórmula [24]:

$$N = \left(2z_{\alpha/2} s / w \right)^2$$

Em que *s* é o desvio padrão do espaço amostral e *w* é o intervalo de confiança das amostras válidas. Para uma distribuição normal e nível de confiança de 95%, o valor de $z_{\alpha/2}$ é de 1,96.

A variável escolhida para o cálculo do número de execuções foi o tempo médio de execução das aplicações. Como não conhecíamos o desvio padrão e o intervalo de confiança da amostra, realizamos 20 execuções de cada cenário, utilizando o esquema de contabilidade perfeita, e calculamos o desvio padrão e o intervalo de confiança para o cenário. Após isso, calculamos o número de simulações necessárias para cada cenário (ver Tabela 3). Como os cálculos mostraram que o maior número de execuções necessárias para alcançar nosso intervalo de confiança foi 280, resolvemos executar 280 simulações para cada cenário.

	Número de Execuções
Cenário 1	238
Cenário 2	263
Cenário 3	276
Cenário 4	280
Cenário 5	279
Cenário 6	277
Cenário 7	222

Tabela 3. Número de simulações necessárias para cada cenário.

3.5 Execução das Simulações na Grade

Após o cálculo do número de amostras necessárias, geramos as instâncias de cada cenário e utilizamos cada instância gerada como entrada para a execução com os esquemas de: (i) contabilidade perfeita; (ii) usando tempo e (iii) utilizando poder relativo.

A geração de uma instância de cenário em um Pentium IV com 640 MB de memória dura em média 25 minutos. Já a execução de cada instância de cenário de simulação dura 35 minutos por esquema de contabilidade. Se não tivéssemos usado a grade para executar as simulações, seriam necessários naquela máquina aproximadamente 205800 minutos ou 3430 horas ou 142 dias em um regime de

execução de 24 horas. Também seriam necessários aproximadamente 34 dias de execuções para gerar os cenários. Nesse tempo não está incluído o tempo para as simulações que apresentarem erros ou cujos resultados não estão apresentados nesta dissertação.

Para contornar o problema do tempo necessário para realizar as simulações nós utilizamos o software OurGrid. O OurGrid é um sistema peer-to-peer que consegue executar aplicações bag-of-tasks. As simulações executadas durante a realização deste trabalho pertencem à classe de aplicações bag-of-tasks por não precisarem de comunicação durante a execução.

A comunidade OurGrid nos forneceu em média 25 máquinas e aumentou o speedup de nossa execução em aproximadamente 12 vezes, fazendo com que o tempo de execução das simulações presentes nesta dissertação (caso fossem somados o tempo de execução das simulações e da geração de cenários) fosse por volta de 15 dias.

3.6 Análise dos Resultados

O método de avaliação dos esquemas de contabilidade por tempo e contabilidade utilizando o poder relativo do peer foi a comparação com um esquema de referência alimentado por informações perfeitas a respeito do poder de processamento das máquinas e custo computacional das tarefas. O objetivo da avaliação foi mensurar o impacto da utilização dos esquemas de contabilidade nos cenários apresentados anteriormente.

A contabilidade utilizando o esquema perfeito é realizada pelo consumidor e pelo provedor. No esquema perfeito, a contabilidade *Acc* do favor *f* é feita da seguinte maneira:

$$Acc(f) = p(máquina) * t(f) = c(T)$$

Onde *p(máquina)* é o poder computacional da máquina, *t(f)* é o tempo durante o qual a máquina foi disponibilizada para execução e *c(T)* é o custo computacional da tarefa *T*.

A utilização de dado esquema de contabilidade influi na cooperação entre os peers da comunidade. O esquema de contabilidade alimenta o sistema de ranking de cada site, e esse ranking determina a preferência que será dada a cada site consumidor pelos provedores. O provedor utiliza o valor fornecido pelo esquema de contabilidade

para diminuir a prioridade dada ao consumidor que recebeu o recurso. Já os consumidores utilizam esse valor para aumentar a prioridade dada ao provedor do qual recebeu esse recurso. Deste modo, deformações no balanço de contabilidades interferem em quais consumidores tem mais prioridade para acessar recursos de provedores em momentos de contenção de recursos, isto é, quando mais de um consumidor requisita recursos ao provedor ao mesmo tempo.

Os valores fornecidos por cada esquema de contabilidade estão em escalas diferentes e por isso não são diretamente comparáveis. Porém, um esquema de contabilidade justo deve produzir um comportamento o mais próximo possível do comportamento produzido pelo esquema perfeito em uma comunidade de compartilhamento de recursos.

Em particular, é necessário avaliar se a comunidade retribui aos peers de acordo com suas contribuições, ou seja, se o esquema de contabilidade contribui para uma comunidade de compartilhamento de recursos justa. É necessário identificar também como os esquemas de contabilidade afetam os usuários que submetem aplicações para a grade. Nesse caso, desejamos verificar se os ganhos no tempo de resposta das aplicações são proporcionais ao do peer local.

As métricas utilizadas para avaliar os esquemas de contabilidade foram: *razão de favores* e *tempo de resposta da aplicação*.

A razão de favores é a razão entre os recursos consumidos pelo peer e os recursos doados para a comunidade. A razão de favores reflete a dinâmica de cooperação entre os peers de uma comunidade. Um esquema de contabilidade justo é aquele em que os resultados diferem o mínimo possível da dinâmica de cooperação alcançada quando um esquema de contabilidade perfeita é utilizado.

[1] avaliou o comportamento da razão de favores em comunidades de peers consumidores e provedores. A avaliação foi realizada levando em consideração que havia contenção de recursos. A rede de favores proposta naquele trabalho objetivou equalizar a quantidade de recursos doados e a quantidade de recursos recebidos pelos peers, ou seja, garantir que a quantidade de recursos doados por cada peer à comunidade fosse próxima da quantidade de recursos recebida da comunidade. Mas note que os estudos realizados em [1] assumem a existência do esquema de contabilidade perfeita.

Em [1], a rede de favores se mostrou eficiente para comunidades de peers de diversos tamanhos e a tendência da razão de favores foi se aproximar de 1, ou seja, as doações realizadas por cada peer foram proporcionais aos consumos nas comunidades estudadas mesmo na presença de free-riders, que naquele trabalho foram considerados como consumidores que não realizam doações.

Entretanto, na vida real (e em nosso simulador) existe outro fator que influencia a razão de favores, o cancelamento de tarefas. O peer aborta todas as tarefas de outros peers quando uma aplicação local é escalonada para ele. O favor incompleto não tem nenhum valor para os consumidores, logo, os consumidores não dão nenhum crédito aos provedores nesses favores. Então, no nosso simulador, esperamos que a razão de favores convirja para um valor menor que 1.

O cálculo da razão de favores leva em conta as interações dentro da comunidade e é definido como:

$$FR = \frac{\sum favoresConsumidos}{\sum favores\,\mathrm{Re}alizados}$$

É importante ressaltar que, embora os esquemas de contabilidade comparados sejam diferentes, utilizamos o esquema perfeito para contabilizar a razão de favores ao final da simulação para garantir que os resultados possam ser comparados.

A nossa segunda métrica é o tempo de resposta da aplicação. Essa métrica representa o tempo necessário para um peer executar todas as tarefas de uma aplicação submetida a ele. O nosso objetivo é verificar a velocidade com que a comunidade responde às requisições dos peers de acordo com suas contribuições. O tempo de resposta da aplicação *(Tr)* é calculado da seguinte forma:

$$Tr = (Ts - Tf)$$

Onde *Ts* é o tempo no qual a aplicação foi submetida e *Tf* é o tempo no qual a última tarefa da aplicação terminou. A expectativa é que o tempo de resposta das aplicações submetidas a um peer seja inversamente proporcional à quantidade de doações realizadas por ele para a comunidade.

CAPÍTULO 4

Esquemas de Contabilidade Autônoma

Este capítulo descreve os esquemas de contabilidade avaliados nesta dissertação. A ênfase será mostrar os pontos fortes de cada esquema de contabilidade, bem como suas principais vulnerabilidades.

A.1. Contabilidade por Tempo

O esquema de contabilidade por tempo é o mais simples de realizar contabilidade. Nesse esquema, a contabilidade para provedores e consumidores é apenas o tempo durante o qual o recurso esteve disponível para executar tarefas. A vantagem desse esquema é a simplicidade de implementação tanto para provedores de recursos quanto para consumidores, já que essa informação pode ser medida independentemente por ambos. A desvantagem desse esquema é que ele não incentiva os provedores de recursos a disponibilizarem máquinas rápidas na grade. O provimento de uma máquina lenta faz com que o consumidor leve mais tempo para executar suas tarefas do que se um recurso rápido fosse utilizado. A conseqüência disso é que os provedores de máquinas lentas podem ter seus favores contabilizados com um valor maior do que os provedores de máquinas rápidas para a execução da mesma quantidade de trabalho.

Mesmo os peers com máquinas rápidas podem simular a doação de máquinas lentas. Este ataque à contabilidade pode ser realizado pela doação do mesmo recurso para mais de um consumidor ao mesmo tempo. Para o consumidor, é difícil identificar esse tipo de ataque em ambientes descentralizados que envolvam vários domínios administrativos, uma vez que o consumidor não tem controle sobre os recursos do provedor. Em um cenário utilizando contabilidade por tempo, a motivação para um provedor doar o mesmo recurso para mais de um consumidor é o potencial aumento de seu crédito com mais de um peer da comunidade ao mesmo tempo e, por conseguinte, o aumento de suas chances de conseguir recursos da comunidade quando necessário. Logo, os provedores de recursos são bastante motivados a realizar esse tipo de ataque. Os resultados alcançados pelos esquemas de contabilidade nessa situação podem ser vistos no cenário 4.

A.2. Contabilidade Utilizando Poder Relativo do Peer

Os resultados apresentados no capítulo 5 mostram que o tempo de uso do recurso é uma boa métrica para contabilizar recursos em grades peer-to-peer nos cenários onde não há variação na potência média das máquinas dos peers. Entretanto, em cenários em que os peers possam ter máquinas de potência média diferentes, esse esquema de contabilidade pode causar desequilíbrio do sistema, ou seja, fazendo com que o fornecimento de recursos não seja proporcional ao consumo para alguns peers.

Com o objetivo de aumentar o número de cenários em que um esquema de contabilidade autônoma é útil sem comprometer a simplicidade de implementação, nós propomos neste trabalho um esquema de contabilidade autônoma que utiliza a própria execução das tarefas nos recursos da grade para realizar contabilidade. Esse esquema visa a ser justo com todos os participantes do sistema de forma que os peers ganhem recursos da comunidade de acordo com a velocidade dos recursos que doam ao sistema. Outro objetivo é que a utilização desse esquema não assuma que existe confiança mútua entre os participantes da comunidade de compartilhamento de recursos.

Neste trabalho, nós projetamos um esquema de contabilidade no qual os peers calculam o valor dos favores fornecidos por outros peers baseado em suas próprias informações. No nosso esquema, os peers não medem o poder computacional real dos outros peers, ao invés disso, a contabilidade é feita baseada no poder relativo dos outros peers da comunidade em relação ao poder do peer local.

Em nosso modelo, os peers utilizam recursos locais e recursos obtidos da comunidade simultaneamente para atender às requisições locais, em outras palavras, para cada aplicação escalonada, o peer utiliza os recursos que tem acesso direto e pede mais recurso para a comunidade. Nosso objetivo é utilizar esse fato para descobrir quão rápidos são os peers da comunidade em média em relação ao peer local e utilizar essa informação para contabilizar a utilização de recursos.

Uma hipótese assumida em nossa solução é que os peers devem ter recursos locais aptos a executar suas próprias tarefas. Nós precisamos dessa hipótese porque utilizamos o tempo de execução das tarefas nos recursos locais como referência por nosso esquema de contabilidade. Nós achamos que essa exigência é factível na maioria

das grades peer-to-peer existentes, pois as comunidades de trocas de recursos que utilizam redes peer-to-peer geralmente não garantem que o usuário obterá recursos da comunidade em todas as oportunidades, sendo apenas um melhor esforço para isso. Logo, é razoável que os usuários busquem a grade apenas para aumentar o desempenho de suas aplicações. Porém, em nosso modelo, quando já existe uma requisição local executando tarefas em todos os recursos, uma nova requisição local pode não ter recursos locais para executar tarefas de uma aplicação. Quando isso ocorre em nossas simulações, o consumidor contabiliza o uso de recursos utilizando apenas o tempo.

Devido às mudanças no poder computacional dos recursos ao longo do tempo e ao fato do consumidor não ter acesso direto e completo aos recursos obtidos da comunidade, torna-se difícil estimar o valor útil de cada recurso disponível na grade computacional. Então, uma vez que os rankings de colaboradores são construídos por peers, nós defendemos que o valor de utilidade do uso dos recursos seja calculado por peer, ao invés de calculá-lo por recursos individualmente. Neste trabalho nós introduzimos o conceito de *poder relativo do peer (prp)* como sendo a razão de quão rápido um peer estrangeiro é comparado ao peer local, ou seja, a razão da velocidade do peer estrangeiro sobre a velocidade do peer local. Mais precisamente, para contabilizar cada favor *f*, ambos os peers utilizam a seguinte fórmula:

$$Acc(f) = prp(P, P') * t(f)$$

Em que *prp(P,P')* é o poder do peer provedor *P'* em relação ao poder do peer local *P*, e *t(f)* é o tempo de utilização do recurso necessário para realizar o favor *f*, ou seja, executar as tarefas escalonadas pelo peer *P* no recurso provido.

Para estimar o poder relativo do peer, nós medimos a média de tempo gasto pelas tarefas para serem executadas nos recursos locais e dividimos pela média de tempo gasto pelas tarefas da mesma aplicação para executarem nos recursos de cada peer estrangeiro.

Visto que o poder relativo do peer é estimado de maneira autônoma a partir das tarefas e provedores e consumidores são capazes de medir o tempo de uso dos recursos, o esquema de contabilidade é completamente autônomo.

Quando o peer *P* está provendo recursos, o poder relativo é 1, ou seja, a contabilidade é o tempo no qual o recurso esteve disponível para o consumidor estrangeiro.

Quando o peer *P* está consumindo recursos, o poder relativo do peer do provedor *P'* é:

$$prp(P,P') = \overline{Ptarefas} \big/ \overline{P'tarefas}$$

Em que $\overline{Ptarefas}$ é a média do tempo de execução das tarefas da aplicação atual que executaram localmente e $\overline{P'tarefas}$ é a média de tempo de execução das tarefas da aplicação que executaram nos recursos do provedor *P'*.

Mensurar o poder do peer, ao invés de mensurar o poder dos recursos individualmente, simplifica a contabilidade, pois não exige que o peer consumidor verifique em qual recurso do peer provedor a tarefa foi executada. Outra vantagem dessa abordagem é que não é necessário manter o registro do poder dos recursos individualmente ou estimar um valor diferente para o recurso a cada requisição. Avaliar o poder dos recursos não é trivial, uma vez que o desempenho varia de acordo com a aplicação, isto é, um recurso pode ter um desempenho muito bom para as tarefas de uma aplicação, mas não ser adequado para a execução das tarefas de outra. Além disso, contabilizar por peer torna mais fácil lidar com falhas nos recursos ou mudanças no poder computacional.

Por outro lado, como nós confiamos no tempo de execução das tarefas das aplicações, esperamos que esse esquema funcione melhor quando as tarefas que compõem a aplicação tiverem tamanhos parecidos. Quanto maior a heterogeneidade das tarefas, maior será o erro introduzido na contabilidade devido a essa estimativa. Além disso, outra fonte de erro é a própria variação no poder das máquinas da grade.

4.2.1 Exemplo de uso

Para ilustrar o funcionamento do nosso esquema de contabilidade, vamos apresentar a seguir um exemplo hipotético de sua utilização.

Considere que temos três peers A, B e C que fazem parte de uma comunidade de compartilhamento de recursos. Uma aplicação com doze tarefas é escalonada para o peer A, que possui apenas cinco recursos livres, e este faz uma requisição para a comunidade e é respondido pelos peers B e C com seus recursos livres. Podemos assumir que as tarefas foram escalonadas da seguinte forma (Ver Tabela 4):

Número da tarefa	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Peer	A	B	A	A	B	A	B	A	C	C	C	C
Tempo de execução	10	25	15	15	15	10	20	10	5	12	5	10

Tabela 4. Escalonamento hipotético de tarefas nos peers da comunidade

O primeiro passo é calcular a média do tempo de execução das tarefas nos peers:

$$t(A) = (10+15+15+10+10)/5 = 12$$

$$t(B) = (25+15+20)/3 = 20$$

$$t(C) = (5+12+5+10)/4 = 8$$

A partir das médias calculadas anteriormente, o consumidor A pode realizar o cálculo do poder relativo dos provedores B e C conforme a seguir:

$$prp(A,B) = 12/20 = 0{,}6$$

$$prp(A,C) = 12/8 = 1{,}5$$

Isto significa que o peer B é em média mais lento que o A e tem um potencial equivalente a 0,6 do peer A. Já o peer C é mais rápido e tem potencial equivalente a 1,5 vezes o de A. O valor total dos favores calculados por A para B e C é igual a:

$$Acc(favoresB) = 0{,}6 * (10+15+15+10+10) = 36$$

$$Acc(favoresC) = 1{,}5 * (5+12+5+10) = 48$$

No ponto de vista dos provedores B e C, o cálculo do poder relativo é realizado da seguinte forma:

$$prp(B,B) = 20/20 = 1$$

$$prp(C,C) = 8/8 = 1$$

Já a contabilidade total dos favores realizados ao consumidor A pelos peers B e C é igual a:

Para o peer B: $Acc(favoresA) = 1 * (10+15+15+10+10) = 60$

Para o peer C: $Acc(favoresA) = 1 * (5 + 12 + 5 + 10) = 32$

Note que, embora o peer B tenha provido recurso por mais tempo para o peer A do que o peer C, a velocidade dos recursos é menor, e por isso, a contabilidade total dos favores *Acc(favoresB)* dele também é menor. Para os peers provedores, o resultado da contabilidade é apenas o tempo. Outro ponto de nosso esquema, é que os provedores e consumidores não precisam ter o mesmo valor de contabilidade para os mesmos favores.

4.2.2 Contabilidade Por Poder Relativo com Registro

A estimativa do poder relativo do peer apresenta diferenças em relação aos resultados do esquema perfeito, principalmente quando o custo computacional das tarefas e o poder computacional dos recursos variam. Isso acontece porque nesse caso não é possível saber, por exemplo, se a execução rápida de uma tarefa é causada pelo pouco custo computacional da tarefa ou se a tarefa executou em um recurso rápido.

Uma forma que encontramos para atenuar essa diferença entre o comportamento do esquema perfeito e o do utilizando poder relativo foi manter um registro do poder dos peers ao longo do tempo. Para isso, cada peer consumidor mantém um registro para os peers conhecidos com o par <quantidade de tarefas executadas pelo peer provedor, poder relativo do peer>.

Após o término de todos os favores de uma aplicação, o peer consumidor calcula o poder relativo dos peers durante essa aplicação e recupera o registro do poder relativo armazenado ao longo do tempo. Então, é feita uma média dos poderes relativos mantidos no registro ao longo do tempo com o poder relativo calculado para esta aplicação. Essa média é ponderada pela quantidade de tarefas utilizadas para calcular o poder relativo mantido no registro e o poder relativo calculado para a aplicação corrente. Essa média é então utilizada como poder relativo para contabilizar os favores da aplicação corrente. Por fim, o registro é atualizado com o novo poder relativo calculado e a quantidade de tarefas executadas pelo peer provedor é somada com a quantidade de tarefas calculadas na aplicação corrente.

Outro fator importante é que a hipótese do peer local ter recursos para rodar as tarefas de cada aplicação é minimizada a apenas a primeira aplicação executada por um peer provedor. Mantendo o registro do poder relativo da aplicação, o peer consumidor

pode utilizar o poder armazenado no registro quando não tiver recursos locais ao invés de utilizar poder relativo 1 nesse caso.

No próximo capítulo são apresentados os resultados obtidos durante o mestrado na avaliação dos esquemas de contabilidade. Apresentamos os resultados da contabilidade utilizando apenas tempo e os da contabilidade por poder relativo em relação ao esquema de contabilidade alimentado por informações perfeitas. A ênfase das discussões são os pontos fortes e fracos dos esquemas de contabilidade e os cenários nos quais eles mais se adequam.

CAPÍTULO 5

Avaliação dos Esquemas de Contabilidade Autônoma

Para avaliar o esquema de contabilidade por tempo e contabilidade por poder relativo, nós os comparamos com o esquema de contabilidade perfeita. Esse é o esquema de contabilidade em que os peers realizam a contabilidade com a ajuda das informações que alimentam o simulador, como poder computacional das máquinas e custo computacional das tarefas. Note que esse esquema não é implementável na prática, uma vez que essas informações não estão disponíveis para os participantes.

A utilização de um esquema de contabilidade perfeita incentiva os provedores de recurso a colocarem recursos rápidos na grade com o objetivo de obter mais benefícios da comunidade em troca. Nesse caso, o provimento da mesma máquina para mais de um consumidor ao mesmo tempo não traria resultado algum para o provedor.

A seguir, mostramos cenários nos quais avaliamos os esquemas de contabilidade por tempo e contabilidade por poder relativo e a comparação deles com o esquema de contabilidade perfeita. A ordem dos cenários apresentados parte de cenários homogêneos, nos quais a contabilidade é mais simples de ser realizada, para cenários mais heterogêneos, nos quais as diferenças entre os esquemas podem ser vistas de forma mais clara. No cenário 7, que consideramos o mais realista porque tanto as máquinas dos peers quanto as tarefas das aplicações variam, mostraremos os resultados obtidos com a contabilidade por poder relativo com registro.

Cenário 1

No cenário 1, todas as máquinas da grade têm o mesmo poder computacional e todas as tarefas das aplicações têm o mesmo custo computacional. A Figura 8 mostra os histogramas das taxas de favores recebidos sobre os favores fornecidos utilizando o esquema de contabilidade perfeita, contabilidade por tempo e contabilidade por poder relativo para o cenário 1[2]. Os gráficos são idênticos e mostram que os esquemas de contabilidade por tempo e de contabilidade por poder relativo funcionam bem em um

[2] Note que o número de ocorrências de todos os histogramas de razão de favores presentes nesta dissertação é igual a 480, ou seja, 280 (número de instâncias de cenário executado de cada esquema de contabilidade) x 15 (número de peers em cada instância de cenário).

cenário totalmente homogêneo. Nesse cenário, a média da razão de favores foi de 0.9748 para os três esquemas, sendo o desvio padrão da razão de favores igual a 0.0645 tanto no esquema por tempo como no por poder relativo. Note que a razão de favores é menor que 1, uma vez que consideramos que os consumidores não contabilizam o badput, visto que esse trabalho não gerou nenhum resultado útil para eles.

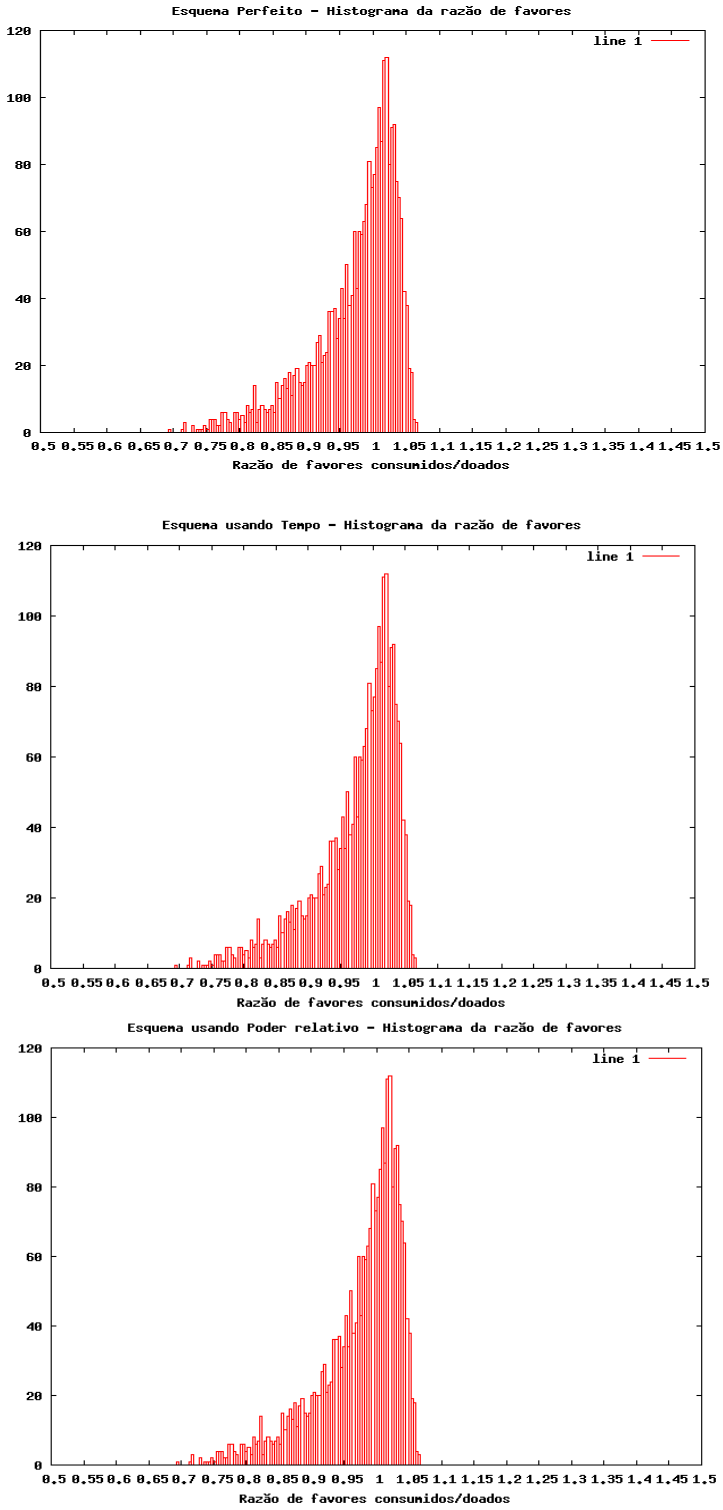

Figura 8. Histogramas da razão de favores para os esquemas de contabilidade no cenário 1.

De fato, para a contabilidade por tempo, esse era o comportamento esperado. Assumindo que x é o poder computacional de todas as máquinas e $t(f_i)$ é o tempo de duração de cada favor consumido f e $t(g_i)$ é o tempo de duração de cada favor provido, em um cenário onde o peer p tenha sido consumidor nos favores $f_1, f_2, f_3,..., f_n$ e tenha sido provedor nos favores $g_1, g_2, g_3,..., g_n$ a razão de favores FR para o peer p utilizando contabilidade perfeita é igual a:

$$FR(p) = \frac{x*t(f_1)+x*t(f_2)+x*t(f_3)+\ldots+x*t(f_n)}{x*t(g_1)+x*t(g_2)+x*t(g_3)+\ldots+x*t(g_n)} = \frac{t(f_1)+t(f_2)+t(f_3)+\ldots+t(f_n)}{t(g_1)+t(g_2)+t(g_3)+\ldots+t(g_n)} = \frac{\sum_1^n t(f_i)}{\sum_1^n t(g_i)}$$

Equação 1. Cálculo da razão de favores do esquema perfeito quando todas as máquinas são iguais.

Nesse caso, como o poder computacional das máquinas é o mesmo, o poder computacional x não interfere na contabilidade dos favores e a razão de favores no esquema de contabilidade perfeita é igual à razão de favores utilizando contabilidade por tempo.

Para a contabilidade por poder relativo, esse resultado também era esperado, pois quando o peer consumidor tem máquinas iguais a do peer provedor e as tarefas são homogêneas, o poder relativo do peer é 1, ou seja, nesse caso a contabilidade por poder relativo é igual à contabilidade por tempo.

Os resultados alcançados na simulação desse cenário evidenciam a eficiência do esquema de reputação baseado na rede de favores, mesmo quando considerado o badput, visto que, em um cenário homogêneo, a razão de favores é próxima a 1.

Cenário 2

No cenário 2, mantivemos o poder das máquinas igual para todos os peers da grade, porém variamos o custos das tarefas submetidas seguindo a distribuição uniforme U(100,700). A mudança no custo computacional das tarefas não afeta a contabilidade por tempo. Como mostramos na Equação 1, quando as máquinas de todos os peers que compõem a grade são iguais, a razão de favores da contabilidade perfeita é igual a da contabilidade por tempo. Os histogramas da razão de favores da Figura 9 e a Tabela 5 mostram que, assim como no cenário anterior, nesse cenário a contabilidade por tempo teve os mesmos resultados da contabilidade perfeita.

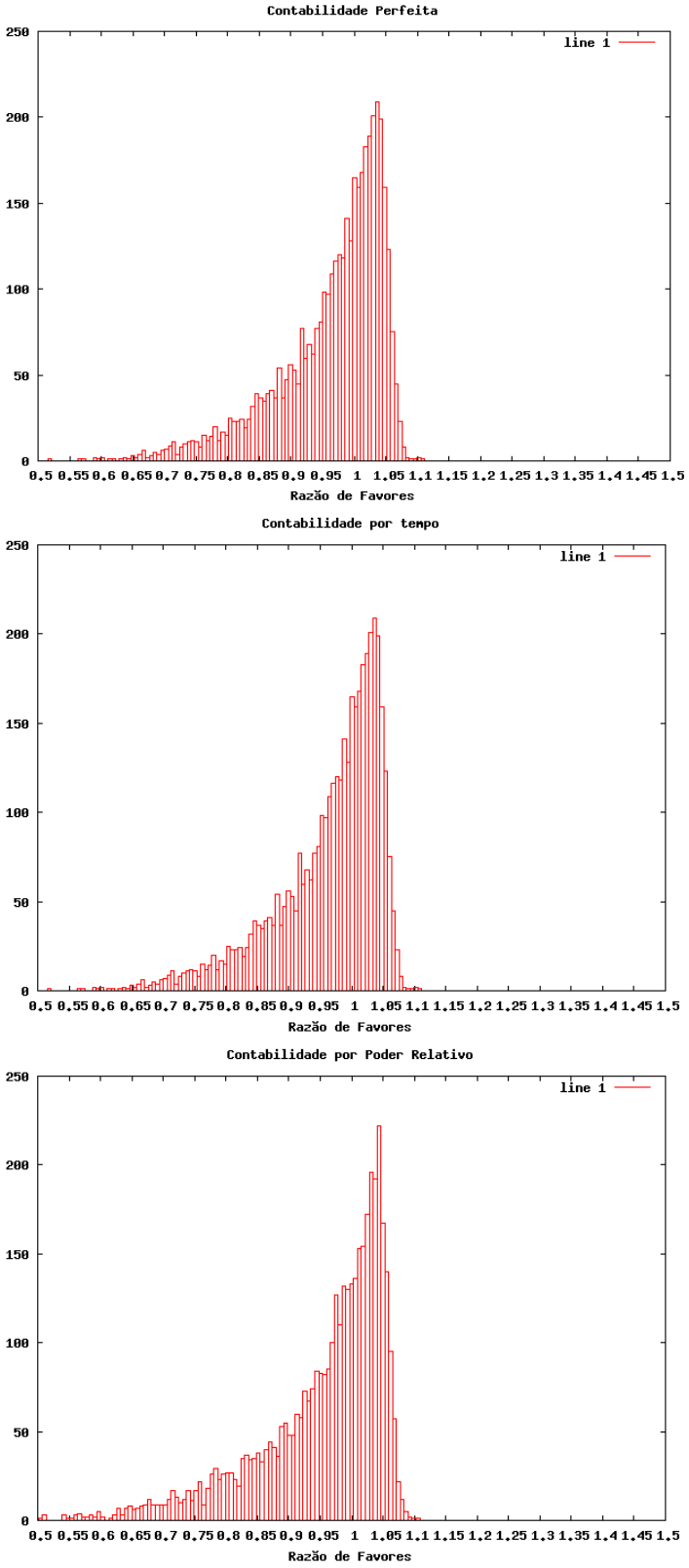

Figura 9. Histogramas da razão de favores para os esquemas de contabilidade no cenário 2.

Nesse cenário, uma vez que a contabilidade por tempo é igual à contabilidade perfeita, esperamos também que a contabilidade por tempo alcance seu melhor desempenho em comparação ao da contabilidade por poder relativo.

Esquema de Contabilidade	Média/Desvio padrão do tempo de resposta	Média/Desvio Padrão da razão de favores
Perfeito	2486.7 / 2,610.7	0.9645 / 0.0851
Tempo	2486.6 / 2,610.7	0.9645 / 0.0851
Poder Relativo	2515.3 / 2,664.6	0.95380 / 0.10197

Tabela 5. Média/Desvio padrão do tempo de resposta e razão de favores no cenário 2.

Os resultados da contabilidade por poder relativo foram um pouco piores que os resultados dos outros dois esquemas. Porém, tanto o desvio padrão da razão de favores quanto o desvio padrão do tempo de resposta das aplicações foram pouco maiores que os alcançados pela contabilidade perfeita e pela contabilidade por tempo. Por exemplo, o desvio padrão da razão de favores para o esquema de contabilidade por poder relativo, que seria grande se alguns peers tivessem a razão de favores muito distante de 1, é apenas 2.5% maior do que o da contabilidade perfeita e o da contabilidade por tempo quando utilizamos contabilidade por poder relativo.

Esse cenário mostra que mesmo no melhor cenário para a contabilidade por tempo, é possível utilizar a contabilidade por poder relativo sem comprometer o incentivo a doações de recursos na grade.

Cenário 3

No cenário 3, mantivemos o custo das tarefas das aplicações constante e variamos a potência média de todas as máquinas dos peers seguindo a distribuição uniforme U (4,16). A Figura 10 mostra que não há diferença significativa nos resultados obtidos utilizando o esquema de contabilidade por tempo em relação ao esquema de contabilidade perfeita em um cenário em que ocorre apenas variação estatística na potência das máquinas dos peers. Isso acontece devido à potência média das máquinas ser estatisticamente igual a 10 para todos os peers. Embora a contabilidade por tempo sofra influência devido a mudanças no poder da máquina, essas mudanças no poder da máquina são compensadas de um peer para outro devido a cada peer ter máquinas rápidas e lentas na mesma proporção dos outros peers da comunidade. O resultado disso é que os desvios padrão da razão de favores e do tempo de resposta são praticamente iguais para a contabilidade perfeita e para a contabilidade por tempo como pode ser visto na Tabela 6.

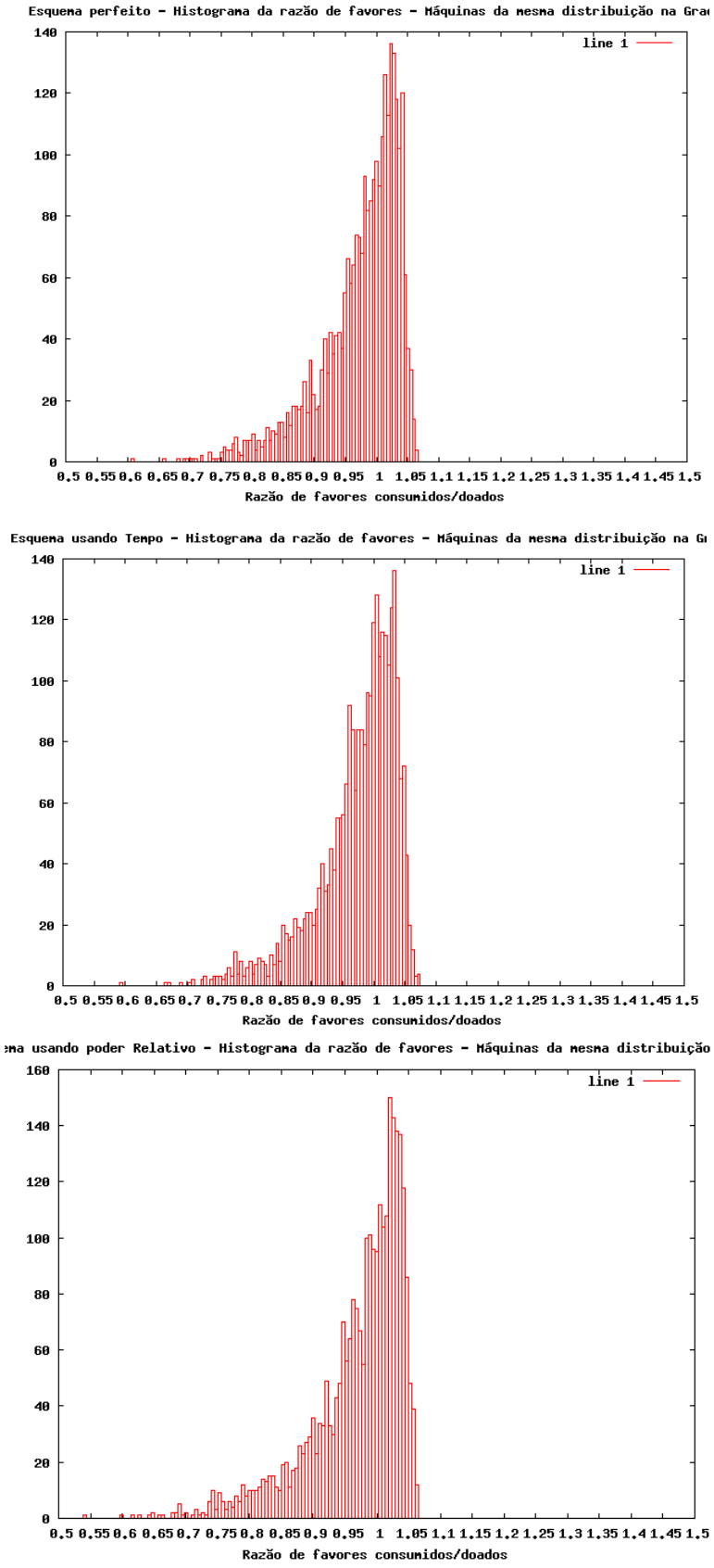

Figura 10. Histogramas da razão de favores para os esquemas de contabilidade no cenário 3.

Também há convergência da taxa de favores consumidos sobre os favores doados para um valor próximo a 1 no caso do esquema de contabilidade por poder relativo. Isso indica que o esquema por poder relativo funciona tão bem quanto os demais quando há apenas variação estatística no poder médio das máquinas entre os peers.

Esquema de contabilidade	Média/desvio padrão do tempo de resposta	Média/desvio padrão da razão de favores
Perfeito	2255.7 / 2296.8	0.97339 / 0.066804
Tempo	2272.3 / 2301.7	0.97251 / 0.067468
Poder Relativo	2277.1 / 2337.6	0.96901 / 0.075126

Tabela 6. Média/Desvio Padrão do tempo de resposta e razão de favores no cenário 3.

Cenário 4

No cenário 4, exploramos o ataque à contabilidade, no qual alguns dos peers executam mais de uma tarefa em cada máquina simultaneamente (peers lentos).

	Média/desvio padrão do tempo de resposta dos peers lentos	Média/desvio padrão do tempo de resposta dos peers rápidos	Média/desvio padrão do tempo de resposta de todos os peers
Perfeito	2940.1/2792.7	2806.7/2690.2	2873.4 / 2742.7
Tempo	2618.9/2637.1	3420.8/2962.8	3019.9 / 2833.2
Poder Relativo	3017.6/2863.8	2619.8 / 2591.9	2818.7 / 2738.5

Tabela 7. Média/Desvio padrão do tempo de resposta das aplicações no cenário 4.

A Tabela 7 mostra que quando um esquema de contabilidade perfeita é utilizado, os peers lentos não obtêm benefícios ao executar duas tarefas no mesmo recurso simultaneamente. De fato, eles estão mais propensos a abortarem tarefas de outros peers porque levam um pouco mais de tempo para processar as tarefas. Isso faz com que eles ganhem menos pelo trabalho realizado, o que reflete em um tempo de resposta um pouco pior quando comparado com os peers rápidos.

Os resultados do tempo de resposta da Tabela 7 mostram que quando a contabilidade por tempo foi utilizada, os peers lentos alcançaram o objetivo do ataque, ou seja, conseguiram mais recursos da comunidade (ver Tabela 9) e com isso reduziram o tempo médio de resposta de suas aplicações. Enquanto a variação do tempo de

resposta das aplicações dos peers rápidos para os peers lentos foi de apenas 4,75% para o esquema perfeito, essa variação foi de -26.44% para o esquema por tempo.

O esquema de contabilidade por poder relativo tem um comportamento muito parecido com o esquema perfeito. Entretanto, a variação do tempo de resposta da contabilidade perfeita para a contabilidade por poder relativo é de 2.61% para os peers lentos e de -6.67% para os peers rápidos. Essa diferença acontece devido ao erro inerente à estimativa de poder relativo, uma vez que não é possível nesse esquema identificar, por exemplo, se os recursos de um peer remoto são lentos, ou se foram escalonadas tarefas com custo maior do que as escalonadas no peer local. Embora a diferença da razão de favores entre peers lentos e rápidos seja maior para a contabilidade por poder relativo do que para a contabilidade perfeita, ela não parece ser grande o suficiente para inibir os provedores lentos de disponibilizarem suas máquinas na grade.

Observando os histogramas de razão de favores da Figura 11, percebe-se que a contabilidade por tempo beneficia os peers que executam duas tarefas simultaneamente em cada máquina também na razão de favores. Observe que, para os peers rápidos, a grande maioria dos peers tem razão de favores inferior a 1, ou seja, doaram mais recursos do que consumiram da comunidade. Já para os peers lentos, mesmo com o badput, existe uma boa parcela que consumiu mais do que doou. A Tabela 8 mostra que a variação da razão de favores ocorrida dos peers rápidos para os lentos utilizando contabilidade por tempo foi de 20,40% enquanto na contabilidade perfeita ela foi de -4,67%. A contabilidade perfeita beneficiou levemente os peers rápidos, porém, pela observação da Tabela 8, percebe-se que no esquema por tempo ocorre o contrário, ou seja, a razão é bem maior para os peers lentos, o que quer dizer que os lentos conseguiram forjar que doaram mais recursos para a comunidade e, com isso, conseguiram consumir mais recursos. A contabilidade por poder relativo também beneficia levemente os provedores rápidos e apresenta resultados bem próximos aos alcançados pela contabilidade perfeita, se mostrando resistente a este ataque à contabilidade também para a razão de favores.

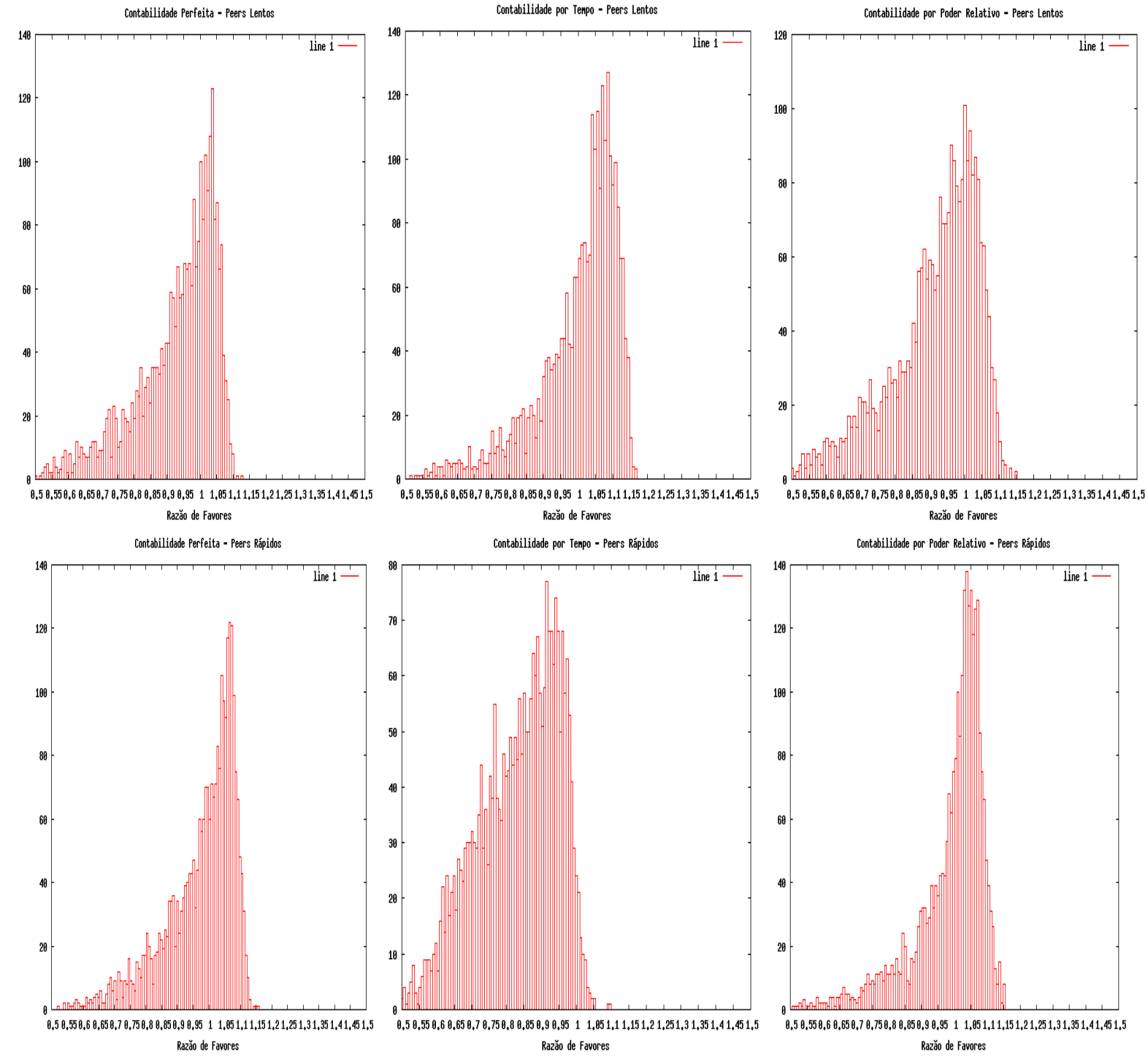

Figura 11. Histograma da razão de favores para esquemas de contabilidade no cenário 4.

	Média/Desvio Padrão da razão de favores para os peers lentos	Média/ Desvio Padrão da razão de favores para os peers rápidos	Média/ Desvio Padrão da razão de favores para todos os peers
Perfeita	0,9304 / 0.1217	0,9760 / 0.1110	0,9532 / 0.1187
Tempo	1,0025 / 0.1166	0,8326 / 0.1171	0,9176 / 0.1445
Poder Relativo	0,9112 / 0.1324	0,9888 / 0.1069	0,9500 / 0.1264

Tabela 8. Média/Desvio padrão da razão de favores no cenário 4.

Outro efeito colateral da contabilidade por tempo é aumentar a participação dos peers lentos nas trocas de recursos da grade (Ver Tabela 9). Uma vez que os peers lentos e rápidos têm o mesmo poder computacional (250), os peers rápidos e lentos deveriam fornecer e receber aproximadamente a mesma quantidade de recursos. Porém, esse comportamento não ocorre com o uso do esquema de contabilidade por tempo. Os peers lentos, como mostram as métricas de razão de favores e tempo de resposta, por aparentarem ter mais máquinas na grade, aumentam seus créditos com os demais peers e com isso ganham prioridade no acesso aos recursos da comunidade. Com mais prioridade no acesso aos recursos, sempre que suas aplicações são escalonadas existe uma probabilidade grande de outros peers doarem recursos para os peers lentos, incluindo recursos que conseguem processar mais tarefas em menos tempo. Nesse cenário, a carga de trabalho recebida por peers lentos e rápidos é a mesma. Então, como os peers lentos conseguem mais recursos do que os peers rápidos da comunidade quando suas aplicações necessitam, os recursos locais dos peers lentos ficam mais tempo ociosos e, portanto, disponíveis para fornecer aos peers da comunidade. Com isso, como mostramos na Tabela 9, os peers lentos conseguem também realizar mais doações de recursos. Por outro lado, para os peers rápidos acontece um comportamento contrário, eles não conseguem muitos recursos, visto que os recursos dos peers da comunidade estão na maioria do tempo sendo utilizados pelos peers lentos. Com isso, os peers rápidos executam a maioria de suas tarefas em seus recursos locais diminuindo sua contribuição com recursos para a comunidade. Note que, para a contabilidade perfeita esse comportamento não ocorre. A quantidade média de recursos dos peers lentos e rápidos é praticamente a mesma, com apenas uma ligeira vantagem para os peers rápidos. O resultado para a contabilidade por poder relativo é semelhante ao resultado do esquema perfeito, porém, devido aos erros na estimativa no poder relativo, os peers lentos participam menos das trocas de recursos que os peers rápidos.

Esse cenário mostra que uma grade peer-to-peer utilizando contabilidade por poder relativo é robusta contra o ataque de executar mais de uma tarefa simultaneamente em cada máquina. A contabilidade por poder relativo disponibiliza aos consumidores informação precisa o suficiente para evitar que os peers provedores forneçam recursos de baixa qualidade e ganhem vantagens contra os peers que fornecem máquinas rápidas.

	Recursos Consumidos/Doados pelos Peers Lentos	**Provedores Consumidos/Doados pelos Peers Rápidos**
Perfeito	6.9059e+06 / 7.4225+06	6.9490e+06/7.1198e+06
Tempo	9.9587e+06 / 9.9338e+06	3.2163e+06/3.8629e+06
Poder Relativo	6.2128e+06/6.8178e+06	7.4012e+06/7.4844e+06

Tabela 9. Recursos consumidos/doados pelos peers no cenário 4.

Cenário 5

No cenário 5, mantivemos o custo computacional das aplicações constante e definimos três classes de peer: com máquinas de potência igual a 4, 10 e 16. A Figura 12 e a Tabela 10 mostram a evolução do tempo de resposta das aplicações relacionada com os tipos de peers definidos para esse cenário. Naturalmente esperaríamos que o tempo médio de resposta das aplicações fosse independente da potência das máquinas dos peers. Porém, com as simulações, pudemos perceber que o tempo médio de resposta aumenta proporcionalmente ao aumento da potência das máquinas dos peers em todos os esquemas avaliados. O aumento existente é devido ao incremento na quantidade de tarefas submetidas para os peers com máquinas de potência maior. Com um maior número de tarefas, aumenta-se a chance de os peers terem as máquinas recebidas de outros peers da comunidade preemptadas durante a execução de suas aplicações, o que causa aumento no badput. Apesar desses efeitos, a alteração na carga de trabalho foi necessária para garantir a saturação dos peers.

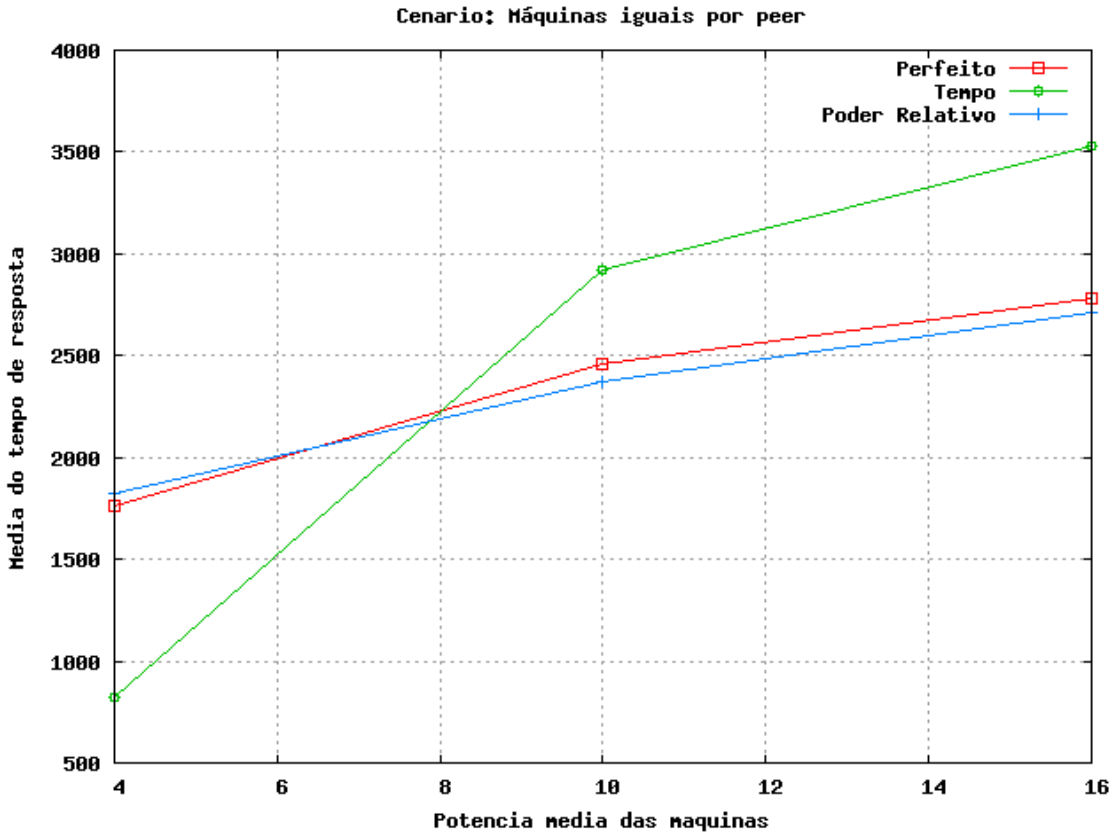

Figura 12. Evolução da média do tempo de resposta das aplicações de acordo com o poder das máquinas do peer no cenário 5.

Pelos resultados da Tabela 10, podemos observar que, para o esquema perfeito, o tempo de resposta médio aumenta 58.1% para os peers com potência média das máquinas igual a 16 em relação aos peers com máquinas de potência igual a 4. Utilizando um esquema baseado apenas no tempo para contabilizar o uso dos recursos, percebemos que o tempo de resposta das aplicações aumenta sensivelmente (329.3%) para os peers com máquinas de poder computacional 16 em relação aos peers com máquinas de poder 4. O tamanho desse aumento indica que os peers com máquinas com menor potência são privilegiados e conseguem mais rapidamente recursos da grade do que quando o esquema de contabilidade perfeita é utilizado.

Esse é o cenário no qual esperamos que a contabilidade por poder relativo tenha os melhores resultados em relação à contabilidade por tempo. Na Figura 12, notamos que a variação do tempo de resposta das aplicações utilizando poder relativo foi quase idêntica à variação do tempo de resposta para a contabilidade perfeita.

Esquema de contabilidade	Média / desvio padrão do tempo de resposta para os peers com máquinas de poder médio 4	Média / desvio padrão do tempo de resposta para os peers com máquinas de poder médio 10	Média / desvio padrão do tempo de resposta para os peers com máquinas de poder médio 16	Média / desvio padrão do tempo de resposta para todos os peers
Perfeito	1758.5/1913.5	2457.3 / 2615.2	2781.6/2829.0	2332.5/2520.0
Tempo	822.01 / 998.03	2921.9/2823.8	3529.6/3106.9	2424.5/2748.3
Poder Relativo	1821.7/ 1889.4	2372.8/2533.4	2711.8 /2767.5	2302.1/2453.0

Tabela 10. Média/desvio Padrão do tempo de resposta no cenário 5.

Além de observar o comportamento das requisições e a resposta da comunidade, é importante verificar se o esquema de contabilidade é justo para todos os peers, independentemente do poder das máquinas que possui. Para isso, vamos mostrar os resultados da análise da razão de favores consumidos sobre os favores realizados para as três classes de peers (com máquinas de poder 4, poder 10 e poder 16). Os gráficos da Figura 13 mostram os histogramas da razão de favores dos peers para os esquemas de contabilidade perfeita, por tempo e por poder relativo nas três classes: poder 4, 10 e 16.

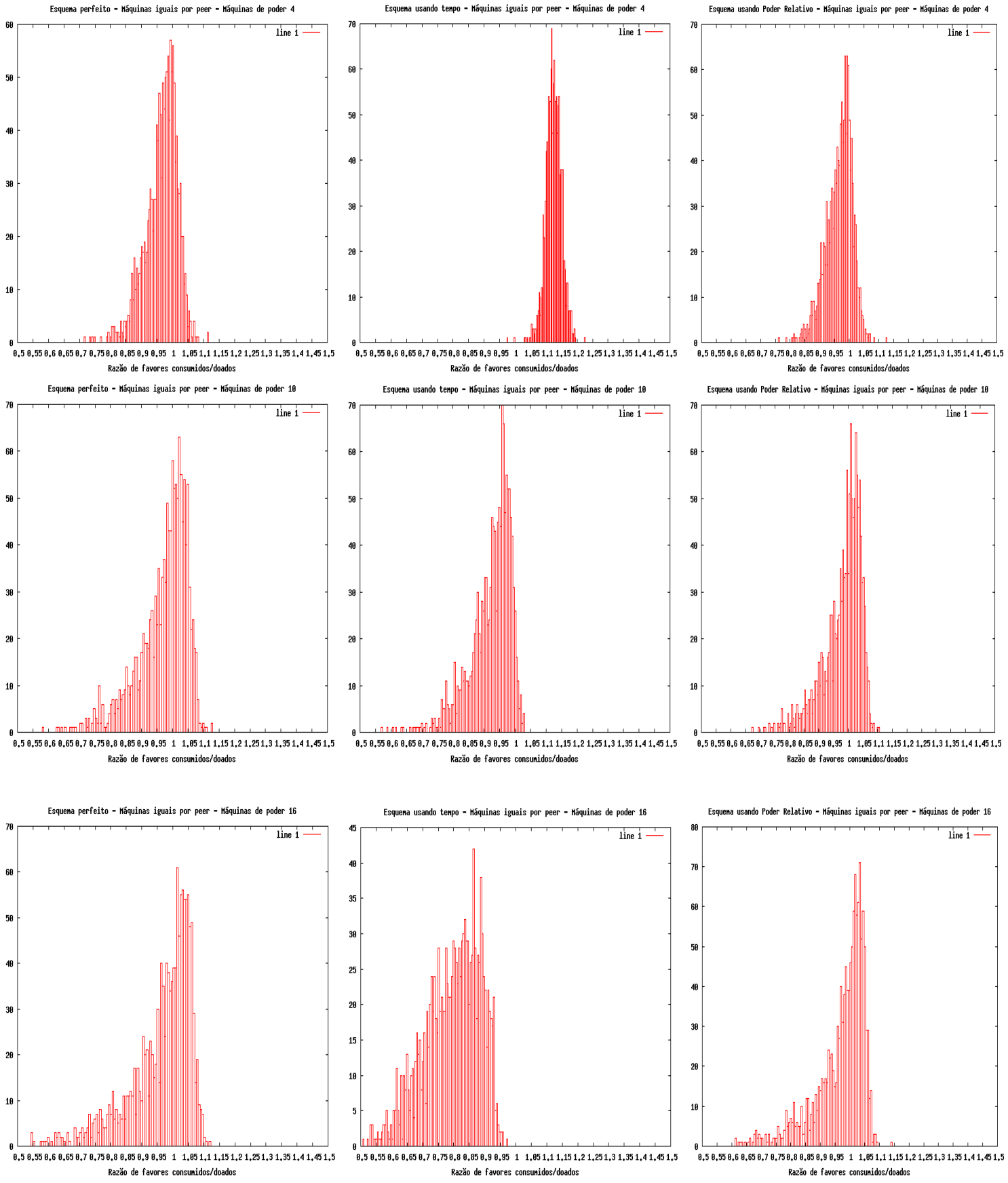

Figura 13. Histogramas da razão dos favores para os esquemas de contabilidade no cenário 5 para os peers das três classes: poder 4, poder 10 e poder 16.

Quando separamos os resultados obtidos pelos peers de acordo com o tipo de máquinas que cada um possui, podemos perceber pelos histogramas que, utilizando um esquema de contabilidade perfeita, o comportamento da razão de favores dos peers é semelhante. Por outro lado, ao analisar os histogramas da razão de favores quando utilizado apenas o tempo como contabilidade, percebemos que os peers que têm máquinas mais lentas se beneficiam mais do sistema. Nenhum peer com máquinas de poder computacional 4 teve a razão de favores consumidos sobre favores realizados menor que 1, ou seja, todos consumiram mais recursos da comunidade do que doaram. Por outro lado, os peers com máquinas mais rápidas tiveram um consumo menor do que a quantidade de recursos doados. Em particular, todos os peers com máquinas de poder computacional 16 doaram mais recursos do que consumiram. Os resultados demonstrados ficam óbvios quando apresentamos a média da razão de favores quando os dois esquemas de contabilidade são utilizados.

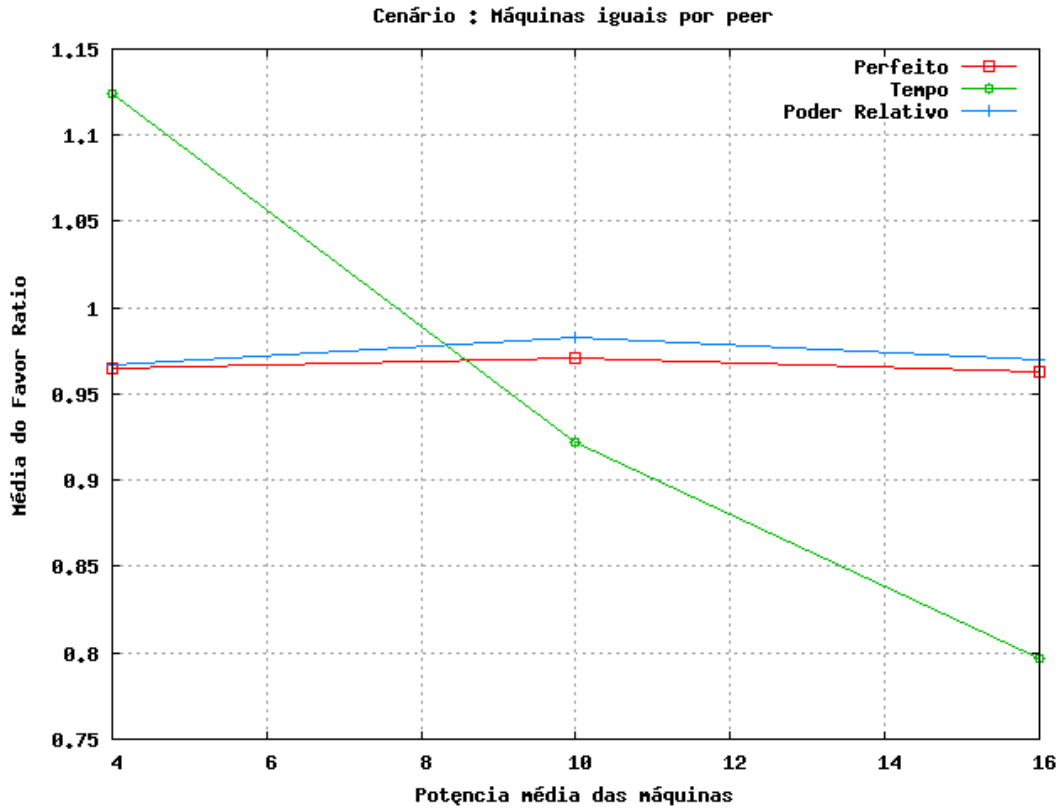

Figura 14. Evolução da média da razão de favores de acordo com o poder das máquinas do peer no cenário 5.

A Figura 14 e a Tabela 11 mostram a evolução da média da razão de favores dos peers, isto é, a razão de recursos consumidos sobre a quantidade de recursos doados por cada participante. Para o esquema de contabilidade perfeita, a razão de favores permanece quase constante, ou seja, mudanças no poder médio não geram efeitos na

dinâmica de troca de recursos dos peers. Quando um esquema de contabilidade baseado no tempo é utilizado, a variação da razão de favores dos peers rápidos para os lentos é de 29.2%. Além disso, o desvio padrão para todos os peers da comunidade é praticamente o dobro no esquema de contabilidade por tempo.

Diferentemente do esquema baseado apenas no tempo, o esquema de contabilidade por poder relativo tem comportamento similar ao comportamento do esquema de contabilidade perfeita. A média da razão de favores consumidos sobre doados permanece próxima a 1 para os peers das três classes.

De fato, nesse cenário, a contabilidade por poder relativo é equivalente à contabilidade perfeita. Assumindo que x é o poder computacional de todas as máquinas do peer P_1 , y é o poder computacional de todas as máquinas de P_2 e $t(f_i)$ é o tempo de duração de cada favor f em um cenário onde o peer P_1 tenha sido consumidor do peer P_2 nos favores f_1, f_2, f_3 ,..., f_n e tenha sido provedor de P_2 nos favores g_4, g_5, g_6,...,g_n a razão de favores FR do peer P_1 utilizando contabilidade perfeita é igual a:

$$FRperfeitoP_1 = \frac{y*t(f_1)+y*t(f_2)+y*t(f_3)+\ldots+y*t(f_n)}{x*t(g_1)+x*t(g_2)+x*t(g_3)+\ldots+x*t(g_n)} = \frac{y*\left(\sum_1^n t(f_i)\right)}{x*\left(\sum_1^n t(g_i)\right)} = \frac{y}{x}*\left(\frac{\sum_1^n t(f_i)}{\sum_1^n t(g_i)}\right) \quad (1)$$

Para calcular a razão de favores de P_1 utilizando a contabilidade por poder relativo, precisamos encontrar o poder relativo de P_2 em relação a P_1. Assumindo que $c(f_i)$ é o custo computacional das tarefas dos favores f_i e que todas as tarefas têm o mesmo custo computacional, o poder relativo pode ser calculado como:

$$prp(P_1,P_2) = \frac{\left(t(f_1)+t(f_2)+t(f_3)+\ldots+t(f_n)\right)/n}{\left(t(g_1)+t(g_2)+t(g_3)+\ldots+t(g_n)\right)/n} = \frac{t(f_1)+t(f_2)+t(f_3)+\ldots+t(f_n)}{t(g_1)+t(g_2)+t(g_3)+\ldots+t(g_n)} \quad (2)$$

Como as tarefas são iguais, todas as tarefas que executarem no peer P_1 terão tempo igual a $c(T)/x$ e as tarefas que executarem no peer P_2 terão tempo igual a $c(T)/y$, em que $c(T)$ é o custo computacional de cada tarefa T, logo:

$$prp(P_1,P_2) = \frac{\left(n*c(T)\right)/x}{\left(n*c(T)\right)/y} = \frac{y}{x} \quad (3)$$

Conhecendo o poder relativo do peer P_2 em relação ao peer P_1 podemos calcular $FRrelativoP_1$, a razão de favores do peer P_1 utilizando poder relativo. O poder relativo de P_2 é utilizado para calcular o valor dos favores consumidos (numerador). Para os

favores realizados de P_1 para P_2 (denominador), o poder é 1, ou seja, apenas o tempo é considerado.

$$FRrelativoP_1 = \frac{\frac{y}{x} * \left(t(f_1) + t(f_2) + t(f_3) + \ldots + t(f_n)\right)}{t(g_1) + t(g_2) + t(g_3) + \ldots + t(g_n)} = \frac{y * (t(f_1) + t(f_2) + t(f_3) + \ldots + t(f_n)}{x} * \frac{1}{t(g_1) + t(g_2) + t(g_3) + \ldots + t(g_n)} =$$

$$= \frac{y}{x} * \left(\frac{\sum_1^n t(f_i)}{\sum_1^n t(g_i} \right)$$

Note que a razão de favores utilizando poder relativo $FRrelativoP_1$ é igual à razão de favores $FRperfeitoP_1$, o que mostra que, nesse cenário, a contabilidade por poder relativo é equivalente à contabilidade perfeita.

A razão de favores do peer P_2 pode ser calculada de maneira análoga a mostrada acima. Existe uma pequena variação das médias da razão de favores em relação ao esquema perfeito (ver Tabela 11). Essa pequena diferença é causada pela utilização do esquema de contabilidade por tempo quando não se tem máquinas locais para executar a aplicação, o que acontece quando já existe uma aplicação local executando em todas as máquinas locais quando uma nova aplicação local é submetida.

Esquema de contabilidade	**Média / desvio padrão da razão de favores dos peers com máquinas de poder médio 4**	**Média / desvio padrão da razão de favores dos peers com máquinas de médio poder médio 10**	**Média / desvio padrão da razão de favores dos peers com máquinas de poder médio 16**	**Média / desvio padrão da razão de favores de todos os peers**
Perfeito	0.96456/0.051731	0.97126/0.080222	0.96246/0.10192	0.96609/0.080691
Tempo	1.1242 / 0.024892	0.92225/0.067078	0.79722/0.089044	0.94790 / 0.15001
Poder Relativo	0.96688/0.044359	0.98246/0.065381	0.97000/0.084883	0.97311/0.067273

Tabela 11. Média/Desvio padrão da razão de favores dos peers no cenário 5.

Cenário 6

No cenário 6, mantivemos os custos computacionais das tarefas constantes e variamos o poder das máquinas que compõem os peers seguindo distribuições uniformes em que a média das potências são 4, 10 e 16. A Figura 15 mostra que, mesmo com peers com máquinas diferentes, as aplicações de peers com máquinas mais rápidas continuam se beneficiando em relação aos peers com máquinas lentas tanto para o esquema perfeito como para o esquema de contabilidade por poder relativo. Já no esquema de contabilidade por tempo, o tempo de resposta médio dos peers com

máquinas rápidas praticamente dobra em relação ao tempo de resposta médio dos peers com máquinas lentas.

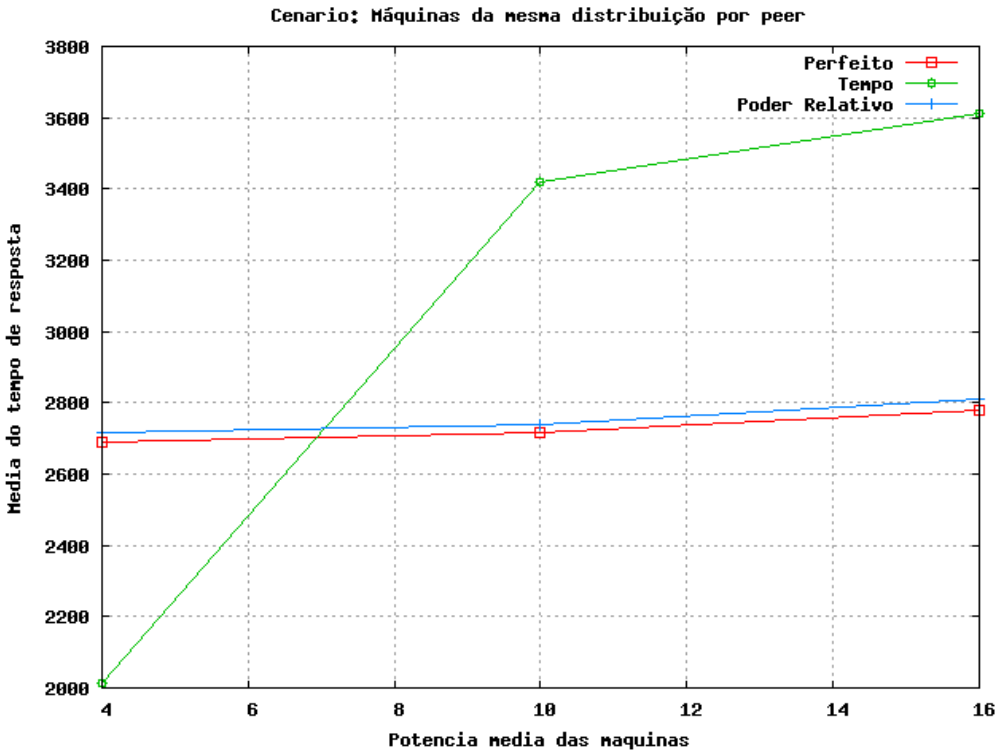

Figura 15. Evolução do tempo de resposta das aplicações de acordo com o poder das máquinas do peer no cenário 6.

Assim como no cenário 5, nesse cenário, o tempo de resposta médio das aplicações é melhor para os peers com máquinas mais lentas quando o esquema de contabilidade por tempo é utilizado. Variando o poder computacional das máquinas dentro do peer, a contabilidade por tempo produz um resultado ainda pior para a razão de favores que no cenário 5 para os peers com máquinas mais rápidas (Ver Figura 16). Isso acontece devido à maior variação do poder das máquinas em relação ao cenário 5. Utilizando o esquema de contabilidade perfeita, a média da razão de favores dos peers apresenta pouca variação. O aumento da razão entre favores consumidos e doados é devido ao menor número de máquinas nos peers com recursos mais rápidos em relação aos peers com recursos lentos, uma vez que, com menos máquinas, mais tarefas são escalonadas em recursos dos peers remotos disponíveis no momento da requisição. Já quando a contabilidade por tempo é utilizada a média da razão de favores decresce quando os peers têm máquinas com potência média maior.

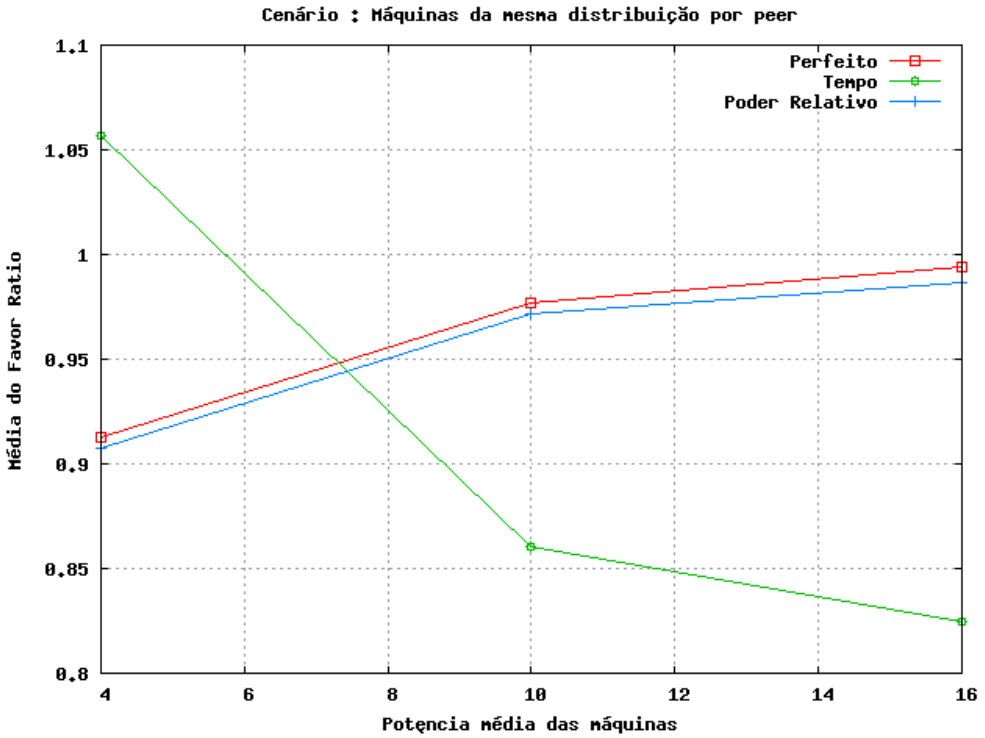

Figura 16. Evolução da média da razão de favores de acordo com o poder das máquinas do peer no cenário 6.

A variação nas máquinas que compõem cada peer não influenciou nos resultados da contabilidade usando poder relativo (ver Tabela 12). Esse caso é ainda mais realista por assumir que existem peers com máquinas mais rápidas e peers com máquinas mais lentas, mas que essas máquinas não são todas iguais. Nesse cenário, fica ainda mais evidente que o esquema baseado no poder relativo apresenta um comportamento mais parecido com o comportamento do esquema perfeito beneficiando levemente os peers com máquinas rápidas, e mantendo as vantagens inerentes a um esquema de contabilidade autônoma.

Esquema de contabilidade	**Média / desvio padrão da razão de favores dos peers com máquinas de poder médio 4**	**Média / desvio padrão da razão de favores dos peers com máquinas de médio poder médio 10**	**Média / desvio padrão da razão de favores dos peers com máquinas de poder médio 16**	**Média / desvio padrão da razão de favores de todos os peers**
Perfeito	0.93118/ 0.11308	0.97721/0.09677	0.99422/0.091382	0.96154/0.10668
Tempo	1.0564/0.079835	0.86034/0.09349	0.82478/0.093550	0.91233/0.16224
Poder Relativo	0.92398/ 0.12602	0.95655/0.12763	0.97713/0.12160	0.95255/0.12700

Tabela 12. Média/desvio padrão da razão de favores dos peers no cenário 6.

Cenário 7

No cenário 7, além de variarmos a distribuição das potências das máquinas como no cenário anterior, variamos também o custo computacional das tarefas das aplicações. Aplicações com tarefas de custos computacionais diferentes podem causar erros de estimativa no poder relativo. Esse erro ocorre quando são submetidas, para o peer local, tarefas com médias de custos computacionais diferentes das submetidas para o peer provedor. Com isso, o peer provedor pode parecer mais rápido que o peer local quando forem escalonadas tarefas com baixo custo computacional em relação às tarefas escalonadas para o peer local ou mais lento caso tarefas com alto custo computacional sejam escalonadas para ele. Especialmente nesse cenário, em que os recursos dos peers também são diferentes, os resultados da contabilidade por poder relativo não foram tão próximos aos da contabilidade perfeita como nos cenários anteriores, como pode ser visto na Figura 17 e na Figura 18. Para atenuar esse problema, inserimos um melhoramento no esquema de contabilidade por poder relativo que mantém o registro dos poderes relativos calculados em execuções de aplicações passadas. Nesse cenário, vamos avaliar a eficiência do esquema por poder relativo com registro e compará-lo com os esquemas de contabilidade perfeita, por tempo e por poder relativo.

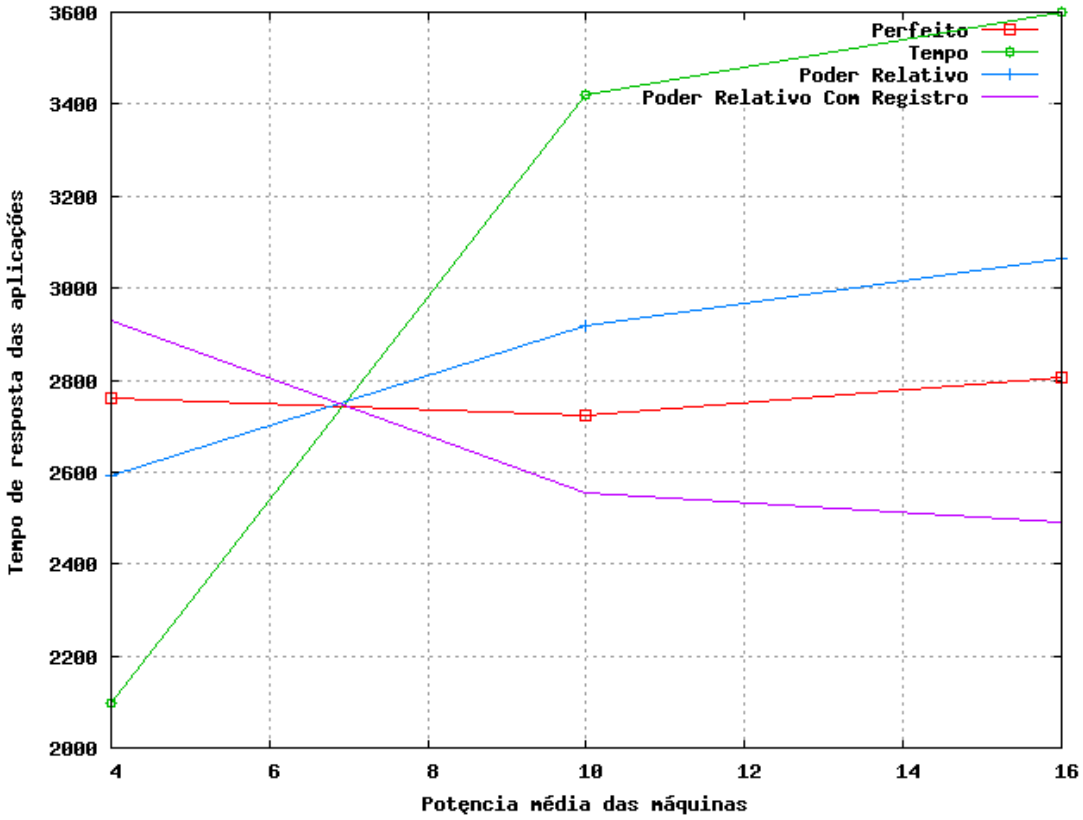

Figura 17. Evolução da média do tempo de resposta das aplicações de acordo com o poder das máquinas do peer no cenário 7.

A variação do tamanho das tarefas não produziu mudanças no tempo de resposta das aplicações para os esquemas perfeito e baseado em tempo. Para o esquema baseado em poder relativo, os resultados pioraram um pouco em relação ao cenário anterior. Porém, mesmo com a influência dos erros de estimativa do poder relativo, o comportamento do tempo de resposta em relação à potência média das máquinas do peer continua sendo bem mais próximo do comportamento do esquema perfeito do que o comportamento do esquema baseado em tempo.

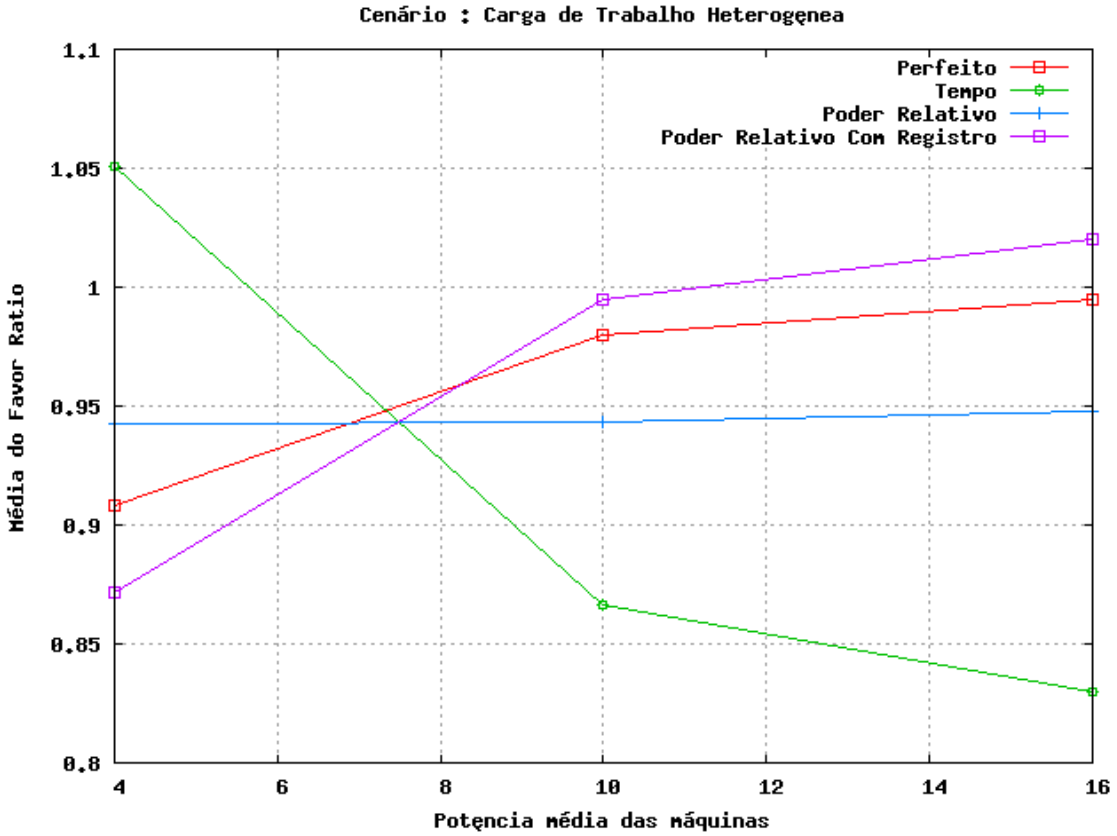

Figura 18. Evolução da média da razão de favores de acordo com o poder das máquinas do peer no cenário 7.

A média da razão de favores consumidos e fornecidos não sofreu influência devido à mudança nas tarefas para os esquemas de contabilidade perfeito e por tempo. Na contabilidade por tempo, a média da razão de favores foi 15.73% maior que a da contabilidade perfeita para os peers com máquinas lentas, poder médio 4, da comunidade. Já para os peers com máquinas de poder médio 16, a média razão de favores foi 16.55% menor para a contabilidade por tempo em relação à contabilidade perfeita. Além disso, o desvio padrão, quando consideramos todos os peers da comunidade, foi 20.08% maior para a contabilidade por tempo em relação à

contabilidade perfeita, ou seja, a diferença entre os peers que se beneficiaram mais da grade e os peers que se beneficiaram menos foi maior.

O esquema de contabilidade utilizando o poder relativo sofreu influência da heterogeneidade das tarefas submetidas em conjunto com as mudanças na carga de trabalho. Porém, mesmo com os erros de estimativa, a diferença entre a contabilidade por poder relativo e a contabilidade perfeita continua pequena tanto nos peers com as máquinas mais lentas (3.74%) quanto nos *peers* com as máquinas mais rápidas (4.72%). Os desvios padrão foram maiores do que os outros dois esquemas em todos os casos, sendo muito próximo da contabilidade por tempo quando consideramos todos os peers.

Utilizando o registro dos poderes passados, o esquema com poder relativo se aproxima ainda mais do comportamento do esquema perfeito. Isso foi causado pela maior quantidade de tarefas utilizadas para calcular o poder relativo dos peers, fazendo com que o erro na estimativa do poder relativo do peer seja menor. Para os peers com máquinas de poder computacional médio 4, a média da razão de favores foi 1.13% maior que a alcançada pelo esquema perfeito, enquanto que para os peers com máquinas de poder médio 16, a média da razão de favores foi apenas 2.51% maior. No geral, a contabilidade por poder relativo com registro teve um desvio padrão muito próximo do da contabilidade perfeita.

Esquema de contabilidade	**Média / desvio padrão da razão de favores dos peers com máquinas de médio poder médio 4**	**Média / desvio padrão da razão de favores dos peers com máquinas de médio poder médio 10**	**Média / desvio padrão da razão de favores dos peers com máquinas de poder médio 16**	**Média / desvio padrão da razão de favores de todos os peers**
Perfeito	0.90816/0.11473	0.98123/0.099193	0.99479/0.10076	0.96139/0.11178
Tempo	1.0510/0.083460	0.86634/0.094554	0.82973/0.10016	0.91568/0.13423
Poder Relativo	0.94247/0.13839	0.94316 / 0.13729	0.94777/0.13950	0.94446/0.13838
Poder Relativo com Registro	0.87191 / 0.12232	0.99494 / 0.079069	1.0201 / 0.075128	0.96233 / 0.11464

Tabela 13. Média/Desvio padrão da razão de favores dos peers no cenário 7.

CAPÍTULO 6

Conclusão

Essa dissertação propôs um esquema de contabilidade autônomo para grades computacionais peer-to-peer nas quais não há confiança entre as partes. Além disso, avaliamos o esquema de contabilidade proposto, a contabilidade por poder relativo, e o comparamos com o esquema de contabilidade por tempo. Para avaliar os esquemas de contabilidade, nós desenvolvemos um simulador para representar o modelo de grade computacional peer-to-peer e para descrever cenários que representassem o uso de nossa solução.

Os esquemas de contabilidade por tempo e contabilidade por poder relativo foram comparados com um esquema de contabilidade de referência, o esquema de contabilidade perfeita em um cenário onde a rede de favores é utilizada para incentivar a troca de recursos justa entre os peers. Embora seja de difícil implementação, pois requer informação oportuna e precisa sobre o poder de todos os recursos disponíveis na grade e sobre as aplicações submetidas, a contabilidade perfeita nos forneceu a base para a avaliação de nossos esquemas de contabilidade através das métricas: tempo de resposta das aplicações e razão de favores.

A nossa avaliação mostrou que esquemas de contabilidade autônomos resolvem o problema de contabilidade de recursos em grades computacionais com simplicidade de implementação.

Mesmo o esquema autônomo mais simples, a contabilidade por tempo, funciona bem em cenários nos quais não há variação entre as máquinas dos peers. Nesses cenários, o esquema de contabilidade utilizando tempo tem resultados similares aos do esquema de contabilidade perfeita, que utilizamos como referência. Os resultados mostram que, em cenários mais heterogêneos, a utilização apenas do tempo como unidade de contabilidade não é suficiente para incentivar uma cooperação justa entre os peers em uma grade peer-to-peer de compartilhamento de recursos computacionais. Nesses cenários, os peers com máquinas com pouca capacidade computacional são beneficiados. Como conseqüência disso, esse esquema é vulnerável a um ataque à contabilidade no qual um provedor de recursos malicioso aloca o mesmo recurso ao

mesmo tempo para mais de um consumidor com o objetivo de aumentar seu crédito com a comunidade.

O esquema de contabilidade por poder relativo se mostrou eficiente nos cenários investigados, tendo os resultados próximos aos do esquema perfeito e sendo simples de implementar e implantar como o esquema por tempo. Especialmente, em nossos cenários mais realistas, a contabilidade por poder relativo apresentou resultados melhores do que a contabilidade por tempo. A contabilidade por poder relativo também se mostrou resistente ao ataque à contabilidade em que os provedores executam mais de uma tarefa por recurso. Neste trabalho, descrevemos também um aperfeiçoamento da contabilidade por poder relativo, no qual utilizamos o registro dos históricos de poder relativo calculados para a execução de aplicações passadas. Essa variação alcançou resultados melhores que a contabilidade por poder relativo nos cenários nos quais variamos o custo computacional das tarefas.

Como trabalhos futuros a esta dissertação, sugerimos:

- Validar os resultados encontrados com o nosso simulador, com o auxílio da implementação do esquema de contabilidade por poder relativo que realizamos no OurGrid, em cenários reais.
- Avaliar o esquema de contabilidade utilizando poder relativo em cenários com replicação de tarefas. Uma forma de minimizar os efeitos colaterais de grandes variações no custo computacional das tarefas é utilizar replicação de tarefas. Replicação de tarefas é uma técnica comum em soluções para grades computacionais como esquemas de tolerância à sabotagem [4] e escalonadores como Work Queue with Replication [5]. Nosso esquema pode se beneficiar da replicação de tarefas para estimar melhor o poder relativo dos peers, uma vez que replicação diminuirá a heterogeneidade do tempo de execução das tarefas das aplicações.
- Avaliar a contabilidade por poder relativo em conjunto com um esquema de detecção de sabotagem como descrito em [4]. O objetivo dessa avaliação seria verificar se o erro inserido pelas tarefas sabotadas que não conseguissem ser identificadas interferiria na contabilidade de recursos.

- Avaliar a contabilidade por poder relativo em um cenário em que haja transferências de dados. No nosso modelo, não consideramos o tempo de transferência das entradas de dados para as aplicações do consumidor para o provedor nem a transferência dos resultados do provedor para o consumidor. Uma vez que a rede na qual estão os recursos locais de um peer é geralmente muito mais rápida que a rede utilizada para a comunicação entre os peers, é necessário avaliar o impacto das transferências de dados nos resultados do esquema de contabilidade por poder relativo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] N. Andrade, W. Cirne, F. Brasileiro, P. Roisenberg. **OurGrid: An Approach for Easily Assembling Grids with Equitable Resource Sharing.** In Proceedings of the 9th JSSPP, Junho de 2003.

[2] Y. S. Kee, H. Casanova, A. Chien. **Realistic Modeling and Synthesis of Resources for Computational Grids.** In Proceedings of SC'04 Pittsburgh, PA, Novembro de 2004.

[3] L. Marchal, Y. Yang, H. Casanova, Y. Robert. **A Realistic Network/Application Model for Scheduling Divisible Loads on Large-Scale Platforms.** In Proceedings of IPDPS'05 Denver, CO, Abril de 2005.

[4] Luis F. G. Sarmenta. **Sabotage-Tolerance Mechanisms for Volunteer Computing Systems.** Future Generation Computer Systems, 2002

[5] W.Cirne, F.Brasileiro, J.Sauvé, N. Andrade, D. Paranhos, E. Santos-Neto, R. Medeiros. **Grid Computing for Bag-of-Task Aplications.** Third IFIP, 2003.

[6] Ali Raza Butt, Rongmei Zhang, Y.Charlie Hu. **A Self-Organizing Flock of Condors.** SuperComputing2003, 2003.

[7] A. Barmouta and R. Buyya. **GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration.** 26th ACSC2003 Adelaide, Austrália, Fevereiro de 2003.

[8] Walfredo Cirne, Francisco Brasileiro, Nazareno Andrade, Lauro Costa, Daniel Paranhos, Elizeu Santos-Neto, César De Rose, Tiago Ferreto, Miranda Mowbray, Roque Scheer, João Jornada. **Scheduling in Bag-of-Task Grids: The PAUÁ Case**. In Proceedings of the 16th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'2004), Outubro de 2004.

[9] R. Buyya, D. Abramson, J. Giddy, H. Stockinger. **Economic Models for Resource Management and Scheduling in Grid Computing.** Special Issue on Grid Computing Environments, The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, 2002.

[10] R.Buyya, D. Abramson, J. Giddy. **Nimrod/G: An architecture for a resource management and scheduling in system in a Global Computational Grid.** In Proceedings of the 4th ICEHPC, China, 2000.

[11] P.Golle and I.Mironov. **Uncheatable Distributed Computation.** Lecture Notes in Computer Science, 2001.

[12] **OurGrid Web Site.**

www.ourgrid.org

[13] Foster, C. Kesselman, J. Nick, and S. Tuecke. **The physiology of the grid: An open grid services architecture for distributed systems integration**. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

[14] L.McGinnis, W.Thigpen, T.Hacker. **Accounting and Accountability for Distributed and Grid Systems.** In Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002.

[15] W.Thigpen, T. Hacker and B.Athey and L.McGinnis. **Distributed Accounting on the Grid.** Proceedings of the 6th JCIS, USA, 2002.

[16] N. Andrade, F. Brasileiro, W.Cirne and M. Mowbray. **Discouraging Free-riding on a Peer-to-Peer Grid.** In Proceedings of HPDC13 – Junho de 2004.

[17] **Site do SETI@home.**

http://setiathome.ssl.berkley.edu

[18] Ian Foster and Adriana Iammitchi. **On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing.** In proceedings of 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), Berkeley, CA, Feb. 2003

[19] Walfredo Cirne, Francisco Brasileiro, Nazareno Andrade, Robson Santos, Alisson Andrade, Reynaldo Novaes e Miranda Mowbray. **Labs of the World, Unite!!!**. Relatório Técnico. Disponível em www.ourgrid.org.

[20] E. Elmroth, P. Gardfjell, O. Mulmo, and T.Sandholm. **An OGSA-Based Bank Service for Grid Accounting Systems.** In J. Wasniewksi et. al. (eds). Applied Parallel Computing. State-of-the-art in Scientific Computing. Springer Verlag, Lecture Notes in Computer Science, 2004. http://www.imm.dtu.dk/~jw/para04/

[21] **Site do Swegrid**. Julho de 2005.

http://www.swegrid.se/

[22] Nazareno Andrade, Francisco Brasileiro, Walfredo Cirne, Miranda Mowbray **Discoraging Free-Riding in a Peer-To-Peer Grid.** In proceedings of the Thirteenth IEEE International Symposium on High-Performance Distributed Computing, Junho 2004.

[23] G. Akerlof. **The market for lemons: quality uncertainty and the market mechanism.** Quarterly Journal of Economics 84 (3), 488-500, 1970.

[24] Soares F., Farias A., Cesar C., **Introdução à Estatística**. Livros Técnicos e Científicos Editora S.A., Rio de Janeiro, 1991.

[25] A. Guarise, R. Piro, A Werbrouck. **DataGrid Accounting System - Architecture v 1.0**.14 de Abril de 2003. Relatório Técnico. Disponível em http://eu-datagrid.web.cern.ch/eu-datagrid/.

[26] **Site do DataGrid**. Julho de 2005

http://eu-datagrid.web.cern.ch/eu-datagrid/

[27] Piro, R., A. Guarise, and A. **An Economy-based Accounting Infrastructure for the DataGrid**. In: Proc. of the 4th International Workshop on Grid Computing (Grid2003). Werbrouck: 2003. Phoenix, Arizona, USA.

[28] A. Butt, R. Zhang, Y. Hu, **A Self-Organizing Flock of Condors**. Supercomputing'2003.

[29] F. Cappello, S. Djilalia, G. Fedak, T. Heraulta, F. Magniettea, V. Nérib and O. Lodygenskyc. **Computing on large-scale distributed systems: XtremWeb architecture, programming models, security, tests and convergence with grid**. Future Generation Computer Systems, 21(3):417-437, March 2005.

[30] V. Lo, D. Zhou, D. Zappala, Y. Liu, and S. Zhao. **Cluster Computing on the Fly: P2P Scheduling of Idle Cycles in the Internet**, IPTPS 2004.

[31] K. Shudo, Y. Tanaka and S. Sekiguchi. **P3: P2P-based Middleware Enabling Transfer and Aggregation of Computational Resources**. 5th GP2PC, May 2005.

[32] E. Adar and B. Huberman. **Free riding in Gnutella**. First Monday 5 (10), October 2000.

[33] D. Hughes, G. Coulson and J. Walkerdine, **Free Riding on Gnutella Revisited: The Bell Tolls**, IEEE Distributed Systems Online, Volume 6, Issue 6, Junho de 2005.

[34] S. Saroiu, P. Gummadi and S. Gribble. **A Measurement Study of Peer-to-Peer File Sharing Systems**. MMCN'02, USA, Janeiro de 2002.

[35] N. Andrade, M. Mowbray, W. Cirne and F. Brasileiro. **When Can an Autonomous Reputation Scheme Discourage Free-riding in a Peer-to-Peer System?** 4th GP2PC, USA, Abril de 2004.

[36] B. Cohem. **Incentives Build Robustness in BitTorrent**. P2P Econ'03, Junho de 2003.

APÊNDICE A

Accurate Autonomous Accounting in Peer-to-Peer Grids

Robson Santos, Alisson Andrade, Walfredo Cirne, Francisco Brasileiro, Nazareno Andrade
Universidade Federal de Campina Grande

Abstract: We here present and evaluate an autonomous accounting scheme that provides accurate results even when the parties (consumer and provider) do not trust each other. Our accounting scheme relies on the observed relative performance among the parties. It is totally autonomous in the sense that it uses only local information, i.e. there is no exchange of information between the parties. This allows for the deployment of the autonomous accounting without requiring any sort of identification infrastructure, such as certificate authorities. The no need of trust or sophisticated infrastructure make our accounting scheme a perfect fit for peer-to-peer grids, which aim to scale much further than traditional grids by allowing free unidentified entry into the grid. Our results show that the proposed scheme performs very close to a perfect accounting scheme whose implementation is infeasible in most systems, including those we target. Although our autonomous accounting scheme was developed to work with OurGrid, it can also be useful for other systems. The basic requirement to use our accounting scheme is that resource consumers must also be resource providers.[3]

Categories and Subject Descriptors
C.2.4 [**Distributed Systems**]: Grid Computing

General Terms
Measurement, economics, performance, experimentation.

Keywords
Accounting, resource usage, reputation model, peer-to-peer grids.

A.1. Introduction

Computational grids have appeared about a decade ago with the promise of delivering great amounts of computational power to parallel applications by enabling the seamless sharing of resources spread among different administrative domains. The grid promise has latter evolved into enabling on-demand access and composition of computational services located throughout the globe.

[3] Aceito para publicação nos anais do 3th International Workshop on Middleware for Grid Computing, Novembro de 2005.

Meanwhile, the first grid systems went into production: TeraGrid, CERN's LCG, OurGrid, Grid3, and many others. This is not to say, however, that all grid problems have been solved. We are still far from that. In particular, we would like to draw the reader's attention to accounting and scalabity; two issues in grid computing that clearly demand further research.

Accounting aims to measure resource usage. A grid provider typically wants to gauge how much resources a given client consumed, either to control such usage or to charge for it. Since a grid comprises multiple administrative domains, this raises a question of trust: should the consumer believe the accounting gauged by the provider? Moreover, finding a metric to represent resource usage is not a simple task, as we can see in Section A.2.

Scalability is also an issue because grids today require complex installation and set-up, and, more importantly, demand human negotiation among the different administrative domains to define how resources are going to be accessed and shared. This is the case because grids aim to provide sophisticated services, typically with some guarantee on what users can expect of such services [18]. Using peer-to-peer techniques has appeared as a promising approach to foster grid scalability [18]. In fact, there are now many peer-to-peer grid projects, e.g. [28] [29] [30] [31], and the first peer-to-peer grids are also going into production. Naturally, peer-to-peer grids do not yet provide services as rich as traditional grids. Application execution (i.e. CPU sharing) appears to be the first service being tackled in peer-to-peer grids. On the other hand, peer-to-peer grids aim to scale for the planet! They try to create a global scale resource-sharing network, so that anyone who requires large amounts of computing power needs just to connect to this network and use the resources that are available.

But notice that accounting becomes an even tougher problem for peer-to-peer grids. In traditional grids, where participants are known and authenticated, there is typically reasonable trust among the administrative domains, and accounting solutions can rely on such trust. On a peer-to-peer grid, on the other hand, participants can freely join the system, without providing a strong link to their real meat-space identities [19]. Therefore, participants are essentially anonymous, and no solutions (including accounting) can be built assuming trust among the parties.

Yet, accounting may be even more important for peer-to-peer grids than it is for traditional ones. This is because free-riding is a common behavior in peer-to-peer systems [32] [33] [34]. A free-rider is a peer that only consumes resources, never contributing back to the system. If there is a non-zero cost for providing resources to the grid, and free-riders get the same benefit from the system as peers that contribute to the system, then it is in the best interest of peers to free-ride and the system may collapse [23]. Coping with free-riding in a peer-to-peer system typically involves some

reciprocation strategy, on which a given peer tries to help peers that have helped it in the past [35] [36]. Therefore, it is crucial for a peer to be able to gauge the contribution another peer provided to it, a requirement that, in a CPU sharing peer-to-peer grid, translates into a strong need for accurate accounting.

We here present an autonomous accounting scheme that relies on the relative compute power of each peer to provide accurate accounting for peer-to-peer grids. We rely on the fact that, in a peer-to-peer grid, resource consumers are also resource providers to autonomously evaluate the relative performance of the parties, and then use this information to normalize the time a resource has being allocated by a provider to a consumer.

Moreover, our accounting scheme is totally autonomous is the sense that it can be fully evaluated using only local information. In fact, the best and simplest way to avoid the need of trust relationship among the parties in the system is autonomously accounting the resource allocations. Our autonomous accounting system does not need a complex infra-structure for its deployment as encrypted certificates, trusted auditor peers, banking or secure e-cash because no accounting information is exchanged among the participants. Moreover, the accounting values are independently maintained by each peer locally.

The rest of this paper is structured as follows. In Section A.2 we describe the accounting problem and discuss some possible approaches to solve it, including the definition of both a perfect and a simplified time-based accounting scheme. We also discuss the difficulties and drawbacks of using these schemes. Section A.3 presents our autonomous scheme that uses the relative computational power of peers to accurately account resource utilization. Then, in Section A.4, we analyze how well our autonomous accounting performs, comparing it against the perfect and the time-based schemes. In Section A.5 we discuss related works. Section A.6 concludes the paper with our final remarks.

A.2. The Resource Accounting Problem

The accounting of a resource allocation is a function of (i) the processing power that the computing resource provided (or used, depending on the viewpoint of who is calculating the allocation's value) is able to deliver for a particular task at a given point in time, and of (ii) the time interval during which the resource is provided (or used). Assuming that $power_p(\tau,t)$ is the amount of processing power a computing resource P is able to deliver to a task □ at a particular instant of time

t, the perfect accounting of running task □ on processor P starting at some time t_i, for the interval of time □t would be $AP(\tau,P) = \int_{t_i}^{t_i+\Delta t} power_P(\tau,t)dt$.

As can be seen, the difficulty in providing accurate accounting lies in determining $power_p(\tau,t)$ for each instant of time t. Assessing $power_p(\tau,t)$ is complicated by two main factors. First, due to inherent characteristics of the tasks that are executed in the grid, the same computing resource may deliver different power to different tasks even when the resource is subject to the same competing load and allocated for the same amount of time. For instance, a resource with a fast CPU but with a slow disk may be more valuable to CPU-intensive tasks than to I/O-intensive ones. Second, this information has to be inferred autonomously by the consumer peer, without relying on information of untrustworthy peers. In particular, the consumer cannot trust the $power_p(\tau,t)$ informed by the provider of P.

Consequently, it is unlikely that such a perfect accounting scheme can be built, unless some restrictions are imposed. Alternatively, one can try to develop practical accounting schemes that are just good approximations to the perfect scheme. A simple approximation would be to consider that all computing resources have the same power at any given time. For instance, one can assume that all resources have power 1 at all times and simply use the time a resource has been made available to a remote peer as the accounting of its resource allocation. Using such time-based scheme, the accounting of running task □ on processor P starting at some time t_i, for the interval of time □t would produce $AT(\tau,p) = \Delta t$.

Since the time interval a resource is made available is easily and autonomously computed by both consumer and provider, this approach can easily be deployed on grids in which peers do not trust each other. However, such a time-based accounting scheme does not take into account the amount of work done by the resource. Thus, peers whose resources take longer to process tasks will get more benefits from the grid than those whose resources are faster and execute the same tasks faster. In Section A.4.4 we discuss the negative impacts that such an approach may bring to the grid as a whole.

A.3. Accurate and Autonomous Accounting

In order to circumvent the problems just described, we have developed a new approach to do accounting that considers the relative power of the peers in the grid. In a peer-to-peer grid, a peer is typically responsible for a site (i.e. resources under a single administrative domain, normally

connected by a LAN). Since reputation models normally rank peers rather than resources [22][35], we make our accounting scheme calculate the accounting per peer.

A peer has local resources that are used to run some tasks of its jobs and, when idle, to run tasks from remote peers' jobs. We take advantage of this fact to define a new metric, named relative power *rpower*, which is used to account for task executions. The relative power of a remote peer is a measurement of how much faster or slower this peer is, when compared to the local peer. That is, $rpower_B(A)$ is the relative power that a peer A is currently able to deliver, as perceived by a peer B. It is estimated by peer B as the average execution time of the tasks B ran locally, divided by the average execution time of the tasks B ran on A. That is, $rpower_B(A) = \frac{mean\{exectime(\tau_B B)\}}{mean\{exectime(\tau_B A)\}}$, where $\{exectime(\tau_B B)\}$ denotes the execution time of the tasks that B ran on B (i.e. locally), whereas $\{exectime(\tau_B A)\}$ denotes the execution time of the tasks that B ran on A. Also, for any peer A, $rpower_A(A) = 1$.

The relative accounting of running task τ from peer B on peer A starting at some time t_i, for the interval of time Δt is defined in B as $AR_B(\tau, B, A) = \Delta t \times rpower_B(A)$ and in A as $AR_A(\tau, B, A) = \Delta t \times rpower_A(A)$. Note that, since $rpower_A(A) = 1$ for any peer A, when A is providing resources to another peer B, it accounts this allocation by simply measuring the amount of time each of its resources are made available to B.

Furthermore, as both the consumer and the provider can measure the time a resource has been in use as well as evaluate their power relative to that of the resource provider, our scheme is completely autonomous and does not require peers to share information. On the other hand, as we rely on the average execution time of tasks, our scheme works better when tasks demand similar amounts of processing. Heterogeneity in execution time of tasks may introduce errors in our accounting. However, as we shall see in Section A.4.4, this error is not significant in the representative settings we have analyzed.

A.4. Performance Evaluation

We have used simulations to analyze the different accounting schemes. We wrote a new event-based simulator for the peer-to-peer computational grid described in Section A.4.1 and implemented the three accounting discussed: perfect[4], time-based and relative accounting schemes.

[4] In our simulations, the perfect accounting of each task execution is equal the computational cost of the task.

A.4.1. Peer-to-Peer Grid Model

Typically, in a peer-to-peer computational grid, each peer represents an administrative domain. Each peer controls the access to a set of local computational resources and plays two roles: (i) it may work as a provider, allowing its (otherwise) idle resources to execute other peers' tasks, and (ii) it may work as a consumer, using its local resources and any idle resources from other peers to execute its own tasks. However, as pointed out before, free-riding is a common behavior in peer-to-peer systems [32] [34].

We thus assume that there is a mechanism to discourage free riding. More precisely, we assume that peers reciprocate resource allocations using the Network of Favors (NoF) [1][16][35]. NoF is an efficient and lightweight reputation-based resource allocation mechanism designed to promote contributions of resources in a peer-to-peer grid [22]. In NoF, the more resources a peer provides, the more likely it is that it will receive resources from other peers when it needs them. It has been used to promote resource sharing in OurGrid [19].

In NoF terminology, resource allocation is called a favor. Resource providers give favors and resource consumers receive them. For a peer A, the reputation ranking of a peer B is represented by the balance of favors they have exchanged, i.e. the amount of favors A has received from B minus the amount of favors A has provided to B. Also, to prevent id-changing attacks5, balances are always kept non-negative. When a peer has idle resources, it provides them to the grid. If two remote peers request these resources, then the remote peer with the highest local balance gets the resources. However, users of the local peer always have priority over those from remote peers. Thus, whenever tasks from local users are submitted, any tasks from remote users are aborted. This simple set of norms makes it in the best interest of peers to provide to the grid as much computational power as they can [22].

Note that NoF is completely decentralized and autonomous, in the sense that there is no exchange of reputation information among peers. In particular, for any 3 peers A, B and C, the reputation of C in A's eyes bears no relation with the reputation of C in B's eyes. The reputation ranking is calculated using only local information about the favors.

However, for the NoF to work well, it assumes an accurate accounting mechanism in the sense that it reflects the actual amount of resources that are consumed and provided by peers. It is important that the accounting is also autonomous, so that the NoF can continue to be deployed

[5] A peer performs an id-changing attack by leaving the system and immediately re-entering it using a new identification, thus (re)setting its balance to zero.

without requiring any special agreements between peers or access to specialized certification services.

A.4.2. Metrics

In order to analyze the simulations, we have defined two metrics: mean response time of jobs and favor ratio. The response time is the time elapsed between the submission of a job and its completion. The relation between the mean response time and the accounting schemes is explained by the way the NoF works. Peers that provide more processing power to the grid obtain more resources when they make a request. Therefore, peers that contribute more should have smaller mean response times for the execution of their jobs.

Another important metric we have used to help us understanding the behavior of the system is the favor ratio $fr(A) = \frac{resourcesConsumed(A)}{resourcesDonated(A)}$, where *resourcesConsumed(A)* is the total processing power consumed by A and *resourcesDonated(A)* is the total processing power provided by a peer A during the simulation, calculated using perfect accounting. Assuming perfect accounting and that the system has contention for resources, the NoF makes favor ratios tend to 1 for all peers in the grid, thus ensuring fairness [22]. However, in reality, it is not possible to implement perfect accounting. Therefore, we are interested on how "implementable" accounting schemes affect the NoF fairness. Note also that, in practice (and in our simulator), another factor that influences the favor ratio is the abortion of tasks. A peer aborts all foreign tasks when it receives a local job. When this happens, the consumer sees no value in the incomplete favor, and the provider gains no credit in its eyes. Therefore, even for perfect accounting, favor ratios tend to converge to a value smaller than 1 in our simulator.

A.4.3. Statistical Sampling

As evaluation scenarios are generated randomly, each setting evaluated must have enough samples for the results to be statistically meaningful. To do that, we chose a confidence level of 95% and estimated the confidence interval based on the results of 20 sample scenarios. Given our confidence level, in the worst case we needed to collect at least 280 samples. To keep it simple, each setting evaluated involved the execution of 280 scenarios.

As it would take too long to execute all simulations in a desktop computer, we used the OurGrid itself to access hundreds of machines in different administrative domains and run the simulations much quicker. Overall, using OurGrid enabled us to run simulations about 40 times faster than using a single machine.

A.4.4. Performance Evaluation

This section presents the results of the experiments we conducted in order to analyze the performance of the three accounting schemes. To do that, we have defined three settings (each corresponding to 280 scenarios simulated). In all scenarios simulated, we ensured that the load submitted to the grid was high enough to create contention in the grid. This is important because if there is a surplus of resources in the grid, then prioritizing peers has no clear effect, as everyone gets the resources they need from the system. As we are interested in observing how the NoF behaves with different accounting schemes, the system must have, on average, more than one peer interested in the resources available. To cause such contention, the inter-arrival time of jobs at peers follow a uniform distribution whose average is calculated by dividing the total amount of work required to execute an average job by the sum of the power of all the resources in an average peer. Due to the probabilistic nature of job arrival, peers are sometimes idle and sometimes need to ask for more resources from the grid.

The first setting describes a grid consisting of peers that contain resources with different powers, to which tasks of different sizes are submitted. This setting corresponds to the most frequent case we expect to find in practice, in which the grid is formed by peers with heterogeneous resources, and jobs vary in size.

The second setting describes a situation in which peers have the same number of resources, those resources have the same power, but tasks vary in size. Although unlikely in practice, this represents a disadvantageous setting for relative accounting, because the variation in the size of tasks introduces errors in the estimation of the relative power of peers. Moreover, since resources have the same power, the time-based approach should have a performance equivalent to the perfect accounting mechanism.

The last setting shows an accounting attack, a situation in which half of the peers behave maliciously. They attack the accounting mechanism by pretending to have twice the number of resources that they actually have by running two tasks per resource simultaneously. This setting analyzes the robustness of the accounting schemes.

Setting 1: Common Case

In this setting we try to capture the heterogeneity of resources and tasks found in a grid. Thus the power of the resources and the size of the tasks vary. However, we keep the total computing power owned by each peer fixed. This means that peers are able to contribute the same amount to the grid and therefore should get the same amount of resources when they make a request. With perfect

accounting, this should is to translate into all peers having similar mean response times and a favor ratio that is close to 1.

We define three types of peers: those with mean resource power 4, 10, and 16. In the simulations we have a grid with 15 peers, 5 peers of each type. The power of a resource in these types is given by the uniform distributions U(2,6), U(5,15) and U(8,24), respectively. In [2], Kee et al. show that it is reasonable to assume that the grid processors follow a uniform distribution. In addition, the sum of the power of all resources in each peer is 250. Note that this implies that peers have different number of resources.

All jobs submitted to the peers in this setting consist of 250 tasks, each task with its size varying according to a uniform distribution U(200,600). (We also experimented with exponentially distributed tasks and obtained similar results.) As to saturate the peers (on average), the inter-arrival time of jobs at each peer follows the uniform distribution U(1, 799), and 1,000 jobs were submitted to each of them.

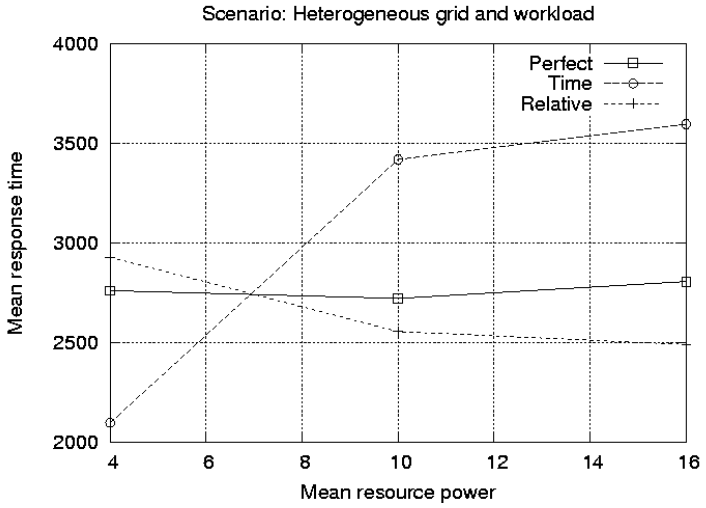

Figure 1. Power vs. mean response time for setting 1

Figure 1 shows the average response time for each type of peer (mean resource power 4, 10 and 16) in the simulations. The fact that peers are able to get the same amount of resources from the grid if everyone accounts correctly is reflected in the very small variation in the mean response times when using the perfect accounting scheme. The reason that there is any variation at all is that peers with slower resources are more prone to abort tasks of other peers running in their resources, as there is a higher probability that local jobs will be submitted during the execution of these tasks.

On the other hand, the time-based accounting is strongly biased towards peers with slower resources. When a resource takes longer to process a task, the favor is (unfairly) assigned a higher

value. Moreover, this difference is very high: peers with slower resources have their jobs processed on average 71.6% faster than those with the fastest resources.

Finally, the relative accounting scheme is slightly biased towards peers with faster resources, but it is much closer to the perfect scheme. Under this scheme peers with faster resources have mean response times 17.6% better than peers with slower ones.

Table 1 shows the favor ratios for each type of peer and each accounting scheme. These results confirm the previous analysis. The mean favor ratio for the perfect and the relative accounting schemes are very close, with the relative accounting scheme being slightly more favorable to peers with faster resources, when compared to the perfect accounting. On the other hand, for the time-based approach there is an enormous discrepancy between the mean favor ratio of the peers with mean resource power 4 and 16, with the peers with slower resources being better off than peers with faster resources.

Table 1. Favor ratio for peers in setting 1

	Mean/Std Dev of *fr* for peers with power 4	**Mean/Std Dev of *fr* for peers with power 10**	**Mean/Std Dev of *fr* for peers with power 16**
Perfect	0.908 / 0.114	0.981 / 0.099	0.994 / 0.101
Relative	0.872 / 0.122	0.995 / 0.079	1.020 / 0.075
Time	1.051 / 0.084	0.866 / 0.095	0.830 / 0.100

This setting gives evidence that the relative accounting scheme is suitable for a peer-to-peer grid, as it has behavior close to that of a perfect accounting scheme. Moreover, the small bias towards peers with faster resources has the side-effect of providing an incentive for peers to make the fastest possible resources available. Since the bias is not large, we believe it should not be enough to put off peers that cannot afford to own fast resources.

Setting 2: Unfavorable Case

In this setting, we have 15 peers of the same type. Each peer has 25 computational resources, and all resources have the same power, which is 10. However, the jobs submitted to each peer contain tasks that are very heterogeneous in size, varying according to an exponential distribution $f(x) = 0{,}0025e^{-0{,}0025*x}$, which has mean equal to 400. Each peer received 1,000 jobs consisting of 250 of those tasks. We created contention in the grid by submitting those jobs with inter-arrival times following a uniform distribution U(1, 799).

This setting is a special case in which accounting a favor correctly is equivalent to accounting the time the task took to be processed in a resource. In such a case, we expect to see the best

performance of the time-based scheme. Moreover, because of the large variation in the size of tasks, this scenario is particularly unfavorable to the relative accounting scheme.

Table 2 shows the response times and favor ratios for each type of accounting scheme. Since all peers in this scenario have the same power and receive the same mean workload, we are here more concerned about the standard deviation of these metrics, in detriment of their means, which should be very close.

As expected, the perfect and the time-based accounting schemes have very similar performance. Considering the response time metric, its standard deviation for the relative accounting scheme is only 3.44% higher than those for the other approaches. When considering the favor ratio metric, the differences are bigger. The standard deviation of the favor ratio (which will be large if some peers have favor ratios far from 1) is 40% higher for the relative accounting scheme than those for the perfect and time-based schemes.

Table 2. Response time and favor ratios for peers in setting 2

	Mean/Std Dev of Response Time	**Mean/Std Dev of Favor Ratio**
Perfect	2,921.6 / 2,783.6	0.9524 / 0.125
Relative	2,951.2 / 2,879.4	0.9193 / 0.175
Time	2,922.1 / 2,782.6	0.9523 / 0.125

We note, however, that most applications we are aware of, are not formed by tasks whose sizes can be modeled by an exponential distribution. They tend to have tasks whose sizes are more uniformly distributed, instead. Our experiments for this scenario considering tasks' size following a uniform distribution U(100, 700) show similar results for the response time metric, whereas the standard deviation of the favor ratio is only 14% higher for the relative accounting scheme when compared to those of the other schemes.

Setting 3: Accounting Attack

This setting gauges the behavior of the accounting schemes when peers try to take advantage of the system by providing resources of poor quality. It is important that consumer peers are able to identify these resources. If providers can receive the same credit, no matter whether the resource is fast or slow, they have an incentive to provide the slowest possible ones, and the system collapses, a phenomenon known as the market for lemons [23].

The setting is a grid with 20 peers, each with 25 machines whose power varies following the uniform distribution U(4,16). However, 10 of such peers run 2 tasks on each machine, appearing for

the grid to have 50 machines of power U(2,8)[6]. We name these peers slower, whereas the other 10 peers, which run 1 task per machine, are called faster. Jobs arrive every U(1,799) time units, each carrying 250 tasks, each task demanding U(200,600) time units to execute.

As Figure 2 shows, when the perfect accounting is used, the slower peers get no benefit from their behavior. In fact, as they take longer to process tasks, they are more prone to abort tasks from other peers. This makes them get less credit in the grid, which is reflected in their slightly worse mean response time when compared to the faster peers.

The relative accounting scheme has similar behavior. However, the difference between the mean response time of the faster and slower peers is a little larger. The mean response time for faster peers is 7.13% shorter than when using the perfect accounting scheme. The mean response time for the slower peers, on the other hand, is 2.63% longer.

For the time-based accounting, we see that slower peers get an advantage when compared to the faster peers, creating an incentive for peers to adopt this strategy. The time-based approach gives two advantages to the slower peers. First, it enables the malicious peer to run twice the number of tasks non-malicious peers can run simultaneously and thus to provide more favors to other peers. Second, it increases the execution time of each task, which benefits the malicious peer, because other peers value its favors based on how long it takes to run their tasks.

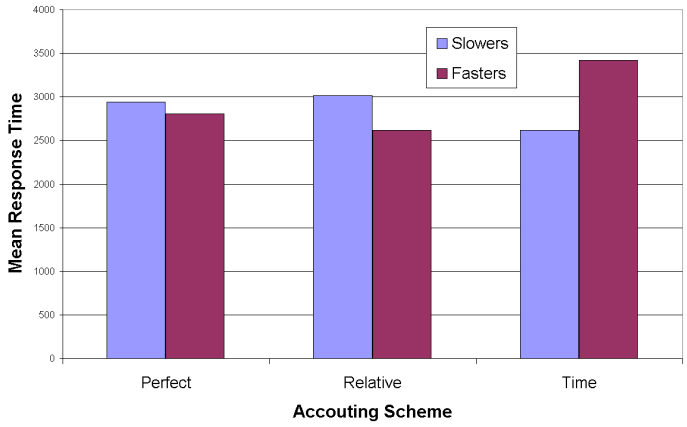

Figure 2. Mean response time for setting 3

Looking at the favor ratios in Table 3 we see how the time-based accounting makes the grid prioritize the slower peers. They get more for the resources they provide to the grid. Again, the relative accounting scheme gets very close to the perfect one, with a standard deviation of favor ratio for all peers only 0.0077 larger than that for the perfect scheme.

---

[6] In practice this problem is exacerbated by the fact that the strategy followed by the malicious peers may introduce trashing in a resource when the tasks that share the resource cannot all fit in

Table 3. Mean favor ratio for peers in setting 3

	Mean/Std Dev of *fr* for slower peers	**Mean/Std Dev of *fr* for faster peers**	**Mean/Std Dev of *fr* for all peers**
Perfect	0.931 / 0.122	0.976 / 0.111	0.953 / 0.119
Relative	0.911 / 0.132	0.989 / 0.107	0.950 / 0.126
Time	1.003 / 0.117	0.833 / 0.117	0.918 / 0.145

This setting shows that using the relative accounting scheme makes the peer-to-peer grid robust against the problem of the market for lemons. It provides consumers with accurate enough information to prevent provider peers serving low-quality resources from gaining an advantage over those providing high-quality ones. Although the relative accounting scheme introduces a larger difference in the favor ratio mean between faster and slower peers than a perfect scheme, this difference (8.5%) is not significantly high. Since it is infeasible to deploy a perfect accounting scheme in a peer-to-peer grid, we believe that the relative accounting scheme is a good enough approximation.

A.5. Related Work

The problem of accounting on grids has been considered before. The major difference between our solution and previous works is that we specifically target accounting among parties that do not trust each other, whereas other proposals assume that parties exchange correct measured data.

In fact, in order to measure the resource utilization and power of resources, GridBank [7] and the DataGrid Accounting System (DGAS)[25] use resource meter agents deployed throughout the grid. We argue that deploying the functionality necessary for collecting utilization data faces serious trust issues for peer-to-peer grids. SweGrid Accounting System (SGAS) [20] uses only the time that the resources are available to run applications as accounting metric. In SGAS case, the accounting is simplified because all resources are equal. However, as we showed in our results, for a more heterogeneous grid, using only time, this benefits providers of slow resources. In short, in GridBank, SGAS and DGAS the accounting is done in the provider side and sent to the consumer or a third-party bank. In a peer-to-peer grid, free-riding providers can send fraudulent accounting information, misleading the consumer. To circumvent this, we autonomously account the resource allocations, so there is no need of exchanging information among parties.

Also, Thigpen et al. [15], DGAS and GridBank assume that there is a previous negotiation between the consumer and the provider before the resource utilization. To help the negotiation, they mention the need to estimate the cost for running a job prior to the use of the resource. This

---

main memory, thus decreasing the power of the grid as a whole.

estimation could be supplied by the user or estimated using historical information. Instead, in our scheme, consumers and providers do not negotiate before execution and autonomously estimate the relative peer power during the job execution, what greatly simplifies the use of an accounting scheme from the user's point of view.

A.6. Conclusions

We have analyzed three accounting schemes to support the implementation of an autonomous reputation-based allocation mechanism for peer-to-peer grids. The perfect scheme requires timely and accurate information on the power that all resources in the grid are able to deliver at any given time. It is very unlikely that such a scheme can be implemented in practical peer-to-peer grid systems. The time-based scheme, although very simple to implement and deploy, always gives a noticeable benefit for peers with slower resources in detriment to those with faster resources. As a result, this accounting scheme is vulnerable to a simple attack in which malicious peers voluntarily reduce the power of their resources by executing more than one task at a time in each resource.

The relative accounting scheme behaves closer to the perfect accounting scheme and is also very simple to implement and deploy. Even for the less favorable (unlikely in practice) setting in which the computing resources that form the grid are homogeneous and the workload submitted to the grid is heterogeneous (see Section A.4.4), our experiments have shown that the relative accounting scheme is only slightly worse than the perfect accounting. The main difference when comparing our accounting scheme against the perfect accounting scheme is that it is slightly biased towards peers with faster resources. We believe, however, that this will not have a negative impact on the grid. On the contrary, it makes in the best interest of the peers to provide their most powerful resources to the grid. Peers therefore have an incentive to increase their local power and as a consequence the power of the grid as a whole.