Provendo Eficiência e Justiça em um Sistema de Cache em Disco para Grades Computacionais Entre-Pares

Rafael Silva, Francisco Brasileiro, Raquel Lopes

Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Laboratório de Sistemas Distribuídos
Av. Aprígio Veloso, s/n, Bodocongó
58.429-900, Campina Grande, PB, Brasil

{rafael,fubica,raquel}@dsc.ufcg.edu.br

Abstract. *Computational peer-to-peer (P2P) grids are a cheap and efficient platform to run applications such as "bag-of-tasks", consisting of a large number of tasks that do not need to communicate amongst themselves and can be executed independently. When these applications handle a large amount of data, it is crucial that the overhead involved with data transfer does not impact application performance. Fortunately, many of these applications have a high rate of data reuse, whether in the same or in successive runs of the application, which allows a caching strategy to be used, so to minimize the loss of performance related to data movement. However, implementing a caching system atop non-dedicated resources and subject to usage policies that are not necessarily homogeneous is not a trivial task. In this paper we propose a disk cache mechanism that promotes collaboration amongst the peers forming the P2P grid and results in a fair and efficient allocation of the disk space provided by the peers.*

Resumo. *Grades computacionais entre-pares (P2P, do inglês peer-to-peer) são uma infra-estrutura barata e eficiente para a execução de aplicações do tipo "saco-de-tarefas", compostas por um grande número de tarefas que não precisam se comunicar entre si e podem ser executadas independentemente umas das outras. Quando essas aplicações manipulam uma grande quantidade de dados, é crucial que a sobrecarga envolvida com a transferência de dados não impacte o desempenho da aplicação. Felizmente, muitas dessas aplicações apresentam um alto percentual de reuso de dados, seja em uma mesma execução ou em execuções sucessivas, o que permite que uma estratégia de cache seja utilizada para minimizar as perdas de desempenho relacionadas com a movimentação dos dados. Entretanto, implementar um sistema de cache sobre recursos não dedicados e sujeitos a políticas de utilização não necessariamente homogêneas, não é uma tarefa trivial. Neste trabalho nós propomos um mecanismo de cache em disco que incentiva a colaboração entre os nós que compõem a grade P2P e resulta em uma alocação justa e eficiente do espaço em disco disponibilizado pelos nós.*

1. Introdução

As grades computacionais surgiram em meados da década passada como um paradigma de computação de alto desempenho para aplicações científicas e de engenharia, através do compartilhamento de recursos heterogêneos distribuídos geograficamente, ultrapassando limites de domínios administrativos [Foster et al. 2002]. Comumente, as grades computacionais provêem serviços de processamento paralelo remoto, que pode

ser compreendido como a execução de aplicações em ambientes computacionais que não sejam recursos do próprio domínio de onde está partindo a submissão da aplicação.

As grades computacionais entre-pares (P2P, do inglês *peer-to-peer*), são caracterizadas pela livre adesão de novos nós (*peers*) para doação/uso de ciclos ociosos de CPU. Neste trabalho, consideramos uma grade computacional em que cada domínio administrativo é representado por um nó na grade P2P. Esse nó é responsável pela gerência dos recursos locais (processadores e disco) e pela intermediação de recursos remotos, quando a demanda dos usuários locais excede a oferta de recursos locais.

Grades P2P apresentam-se como uma solução barata e eficiente para a execução de aplicações do tipo "saco-de-tarefas" (BoT, do inglês *Bag-of-Tasks*), compostas por um grande número de tarefas que não precisam se comunicar entre si e podem ser executadas independentemente umas das outras. Dentre essas aplicações, existem diversas que geram e/ou consomem uma expressiva quantidade de dados, conhecidas como aplicações *data-intensive*. A execução eficiente desse tipo de aplicação em grades P2P é limitada por dois aspectos principais: *(i) pequena largura de banda* entre os nós que compõem a grade e *(ii) espaço em disco insuficiente* nestes nós para o armazenamento simultâneo dos dados de todas as aplicações sendo executadas.

Os nós que compõem a grade computacional P2P estão comumente conectados por canais de comunicação de longa distância, caracterizados por uma largura de banda pequena, especialmente se comparada às larguras de banda disponíveis em ambientes de redes locais. Sendo assim, a comunicação entre os nós de uma grade computacional P2P pode ser lenta e o tempo necessário para transportar dados entre nós de domínios administrativos diferentes pode ser equivalente a uma parcela considerável do tempo gasto para processar as tarefas que consomem esses dados. Desta forma, os ganhos obtidos pela paralelização das aplicações podem ser neutralizados pelo tempo necessário para transferir os dados a serem consumidos. Quanto ao espaço disponível para armazenamento, é possível que não haja espaço em disco suficiente para armazenar o dado em outros domínios, sendo a execução remota negada.

Felizmente, diversas aplicações *data-intensive* possuem um elevado grau de reuso de dados em um mesmo conjunto de execuções [Verleyen et al. 2009, Parada et al. 2009] e/ou em execuções sucessivas [Silva et al. 2007]. Neste trabalho nós denominamos esses padrões de reuso *intra-job* e *inter-jobs*, respectivamente. Desta forma, as limitações de banda podem ser minimizadas através da persistência dos dados nos nós de execução, evitando que dados que serão utilizados em futuras execuções sejam novamente transferidos. Nesse sentido, um sistema de *cache* em disco pode viabilizar a execução eficiente de aplicações *data-intensive* em grades P2P. Entretanto, no contexto de grades computacionais P2P, o armazenamento, assim como a computação, obedece a lei do melhor esforço (do inglês *best-effort*), não oferecendo garantia de que o dado persistido estará disponível quando requisitado. Assim, faz-se necessário um mecanismo que incentive os nós a manterem em seus discos os dados transferidos.

Um desafio inerente aos sistemas P2P de *caching* está relacionado ao modo de compartilhamento dos recursos entre os nós. O problema consiste em alocar os espaços de forma justa – proporcional à contribuição dos nós; e eficiente – de forma a maximizar a utilização dos recursos compartilhados. Um segundo desafio a ser superado é incentivar os nós que compõem o sistema a colaborar doando seus recursos. Sem a

colaboração espontânea dos nós, não seria possível compor um sistema eficiente de *cache* em disco, daí a importância deste incentivo. O terceiro desafio é definir um mecanismo de gerência de *cache* em disco capaz de lidar com *free-riders*, que são nós que consomem mas não doam recursos ao sistema. Adar e Huberman foram os primeiros a realizar um estudo mais profundo sobre *free riding* em um sistema de compartilhamento de arquivos P2P [Adar & Huberman 2000]. Eles reportaram que aproximadamente 70% dos nós não compartilhavam dados, e 25% eram responsáveis por 99% da carga do sistema. Este resultado impõe a necessidade de mecanismos capazes de detectar e marginalizar ou punir *free-riders*.

Neste trabalho, apresentamos um mecanismo de gerência descentralizado para um sistema de *cache* em disco para grades computacionais P2P que incentiva a colaboração e resulta em uma alocação justa e eficiente do espaço em disco compartilhado entre os nós marginalizando os *free-riders* em momentos de contenção de recursos[1]. O mecanismo prioriza o armazenamento dos dados das aplicações que são submetidas localmente, cedendo o espaço não utilizado para persistência de dados das aplicações submetidas remotamente. A decisão sobre a divisão do espaço em disco cedido para armazenamento remoto é realizada de forma autônoma por cada nó, sendo baseada na reputação dos outros nós. Cada nó computa um valor de reputação para cada outro nó com o qual ele interage. A reputação de um nó *y* calculada pelo nó *z* é obtida através da contabilização dos serviços ofertados por *y* a *z* e por *z* a *y*. Diferente de outras soluções, como por exemplo, a proposta por Soares et al. [Soares et al. 2008], o mecanismo proposto apenas contabiliza a prestação de um serviço quando um dado previamente armazenado é utilizado durante processamento de uma aplicação que o consome.

É importante ressaltar que este serviço de *caching* foi proposto para dar suporte às aplicações *data-intensive* que consomem uma grande quantidade de dados. Quando a aplicação apenas gera uma grande quantidade de dados, o serviço de *cache* em disco pode ser útil apenas quando esses dados gerados servem de entrada para outras aplicações que também executarão na grade.

Na próxima seção apresentamos uma solução para o problema descrito. A avaliação do mecanismo proposto é apresentada na Seção 3. Na Seção 4 discutimos os trabalhos relacionados apresentando suas deficiências e diferenças em relação ao mecanismo apresentado neste trabalho. Por fim, na Seção 5 são apresentadas as conclusões e direções para trabalhos futuros.

2. Mecanismo de *Caching* Baseado em Reputação

O mecanismo de *cache* em disco apresentado neste trabalho provê uma alocação justa e eficiente do espaço disponível para armazenamento compartilhado entre os nós de uma grade P2P. O espaço em disco doado por um nó prioriza a persistência dos dados das aplicações que são submetidas localmente (dados locais). A *cache* em disco nunca se recusa a armazenar um dado local. A cada dado local armazenado na *cache* em disco é associado um tempo de vida (*lifetime*) que, uma vez expirado, marca o dado para remoção, o que provê mais espaço para armazenamento de dados das aplicações submetidas remotamente (dados remotos). A remoção de dados da *cache* pode seguir

[1] Neste trabalho contenção de recursos significa falta de espaço para persistência de um dado.

qualquer política de remoção de dados, como por exemplo, LRU (*Least Recently Used*) ou LFU (*Least Frequently Used*) [Podlipnig & Böszörmenyl 2003].

Os dados remotos são armazenados nos espaços não utilizados pelos dados locais. Quando há contenção de recursos, a alocação desse espaço será baseada em um mecanismo de priorização inspirado na rede de favores proposta por Andrade et al. [Andrade et al. 2007]. No contexto da *cache* em disco, um favor é definido como a ação de se prover um *byte* de dado para a execução de uma tarefa submetida por um nó. Cada nó mantém o histórico dos favores recebidos e ofertados para cada nó conhecido, sendo um favor recebido contabilizado como positivo e um favor ofertado contabilizado como negativo.

Cada nó tem autonomia para decidir quanto de espaço será alocado para outros nós. A definição do espaço alocado (cota) a um nó é estabelecida a partir do cálculo de uma função de reputação que contabiliza os favores recebidos e ofertados. A contabilidade $B_z^y(t)$ dos favores ofertados e recebidos por um nó z com relação a um nó y em um instante t é definido na Equação 1.

$$B_z^y(t) = max[0, b_z^y(t) - b_y^z(t)] \quad (1)$$

Nesta equação, $b_z^y(t)$ denota a quantidade de dados providos pelo nó y ao nó z para processamento de tarefas até o instante t. $b_y^z(t)$ denota a quantidade de *bytes* providos pelo nó z ao nó y para o processamento de tarefas até o instante t.

Seja K, o subconjunto de nós conhecidos por um nó z, a função de avaliação da reputação $R_z^y(t)$ de um nó y sob o ponto de vista de z em um instante t é definida por:

$$R_z^y(t) = \begin{cases} \dfrac{B_z^y(t)}{\sum_{x \in K} B_z^x(t)}, se \sum_{x \in K} B_z^x(t) \neq 0 \\ 0, caso\ contrário \end{cases} \quad (2)$$

onde $\sum_{x \in K} B_z^x(t)$ representa a diferença entre a quantidade de *bytes* providos e ofertados por todos os nós conhecidos pelo nó z.

Como proposto por Andrade et al., a reputação de um nó nunca se torna negativa. Como a reputação atribuída a novos nós é zero, esse artifício evita ataques do tipo *white-washing*, nos quais um nó sai do sistema e retorna com uma nova identidade a fim de se livrar de uma baixa reputação.

A cota $Q_z^y(t)$ de um nó y em um nó z no instante *t* é computada com base na reputação de *y* em *z* no instante *t* segundo a equação a seguir:

$$Q_z^y(t) = R_z^y(t) \cdot S_z(t) \quad (3)$$

onde $S_z(t)$ denota o espaço total disponível para armazenamento de dados remotos no nó z no instante *t*.

A cota baseada em reputação garante que a quantidade de espaço na *cache* em disco de um nó *y* oferecida para um nó *z* é diretamente proporcional à quantidade de favores que *y* ofereceu a *z* em interações passadas. Desta forma, os nós que compõem a grade computacional são incentivados a oferecer favores e manter em disco arquivos em

uso de nós remotos que aumentam sua reputação em tais nós. À medida que os nós contribuem com a grade eles aumentam sua reputação, e só assim aumentam suas cotas nos demais nós da grade.

Além de incentivar colaborações, a definição de cotas baseada em reputação é uma forma de se marginalizar *free-riders* em prol dos colaboradores. É importante observar que em cenários sem contenção de recursos, a cota não é considerada para alocação de espaço, ou seja, um nó pode ocupar mais espaço do que o estipulado por sua cota. Esta decisão é inspirada no trabalho de Soares et al. [Soares et al. 2008] e foi tomada de forma a maximizar o uso do espaço disponível para a *cache* em disco, mesmo que isto implique em beneficiar *free-riders* em algumas ocasiões. Obviamente, um nó que excede sua cota pode ter dados removidos prematuramente do sistema de *cache* em disco quando o espaço for requisitado por outros nós de maior reputação. A estratégia de remoção dos dados adotada neste trabalho considera a ordem crescente da reputação dos nós, removendo, primeiramente, dados de nós que excederam sua cota e possuem baixa reputação. O Algoritmo 1 detalha esse procedimento.

Algoritmo 1. Armazenamento de um dado remoto

```
01: se (espaçoDisponível ≥ tamanhoDado) então
02:     armazena(dado)
03: senão
04:     se (tamanhoDado ≤ cota(nó_remoto) + espaçoDisponível) então
05:         enquanto (tamanhoDado > espaçoDisponível + espaçoUtilizado(nó)
                      ou há mais dados de nós com cota excedida a remover)
06:             remove dados do próximo nó de menor reputação que esteja com cota excedida
07:             espaçoDisponível ← espaçoDisponível + tamanhoDadoRemovido
08:         fim enquanto
09:         enquanto (tamanhoDado > espaçoDisponível)
10:             remove dados do nó remoto
11:             espaçoDisponível ← espaçoDisponível + tamanhoDadoRemovido
12:         fim enquanto
13:         se (tamanhoDado ≤ espaçoDisponível)
14:             armazena(dado)
15:         senão
16:             rejeita(dado)
17:         fim se
18:     senão
19:         rejeita(dado)
20:     fim se
21: fim se
```

Três características deste algoritmo de armazenamento de dados remotos merecem destaque. Uma delas é que as cotas dos nós podem ser ultrapassadas, desde que apenas dados excedentes de outros nós precisem ser removidos, isto é, dados que ultrapassam a cota desses nós. Uma segunda característica é que ao remover dados excedentes de outros nós inicia-se pelos nós de menor reputação, existindo a chance de não ser necessário apagar dados excedentes de nós de maior reputação. Finalmente, os próprios dados do nó (mesmo que não sejam excedentes) podem ser removidos para dar lugar aos novos dados. Esta abordagem de alocação de recursos permite maximizar a utilização dos recursos compartilhados.

3. Avaliação

A avaliação do mecanismo proposto neste trabalho consiste em demonstrar, através de experimentos de simulações, que *(i)* o mecanismo utiliza o espaço disponível para o sistema de *cache* em disco de forma justa e eficiente e *(ii)* quando o sistema entra em estado de contenção, *free-riders* são marginalizados em favor dos colaboradores.

Nosso modelo de simulação simplifica aspectos de uma grade P2P, como por exemplo, aqueles relativos à topologia e a falhas. Consideramos uma grade P2P composta por um número fixo de nós, com aplicação da rede de favores [Andrade et al. 2007] para distribuição de recursos de processamento e do mecanismo de incentivo à colaboração descrito na Seção 2 para o gerenciamento do espaço de armazenamento compartilhado. O escalonamento das tarefas utiliza a heurística *Storage Affinity*, que associa as tarefas aos processadores de acordo com a afinidade entre eles [Santos-Neto et al. 2004]. A afinidade é proporcional à quantidade de *bytes* requisitada por uma tarefa que já se encontra armazenada no domínio administrativo (*site*) ao qual o processador pertence.

A linha do tempo é divida em turnos. Em cada turno, cada nó submete um conjunto de tarefas BoT para execução na grade. O escalonador de cada *site* associa as tarefas aos recursos de acordo com o grau de afinidade, priorizando a alocação de recursos locais para nós locais. O recurso alocado recebe a tarefa e a executa. Ao término da execução, o recurso alocado registra um favor ofertado ao nó consumidor, enquanto que este contabiliza um favor recebido.

Para mostrar a eficiência e justiça do mecanismo, utilizamos como métrica o nível de satisfação médio $\bar{A}$ entre os nós, definido pela Equação 4.

$$\bar{A}(T) = \frac{BP(T)}{BR(T)} \tag{4}$$

Nesta equação, $BR(T)$ denota a quantidade de *bytes* requeridos pelo conjunto de tarefas submetidas por um conjunto de nós em um turno T, e $BP(T)$ denota a quantidade de *bytes* disponíveis em *cache* e providos para a execução dessas tarefas no turno *T*. Deste modo, $\bar{A}(T) = 1$ indica a situação ideal em que 100% dos dados requeridos estavam disponíveis para execução.

3.1 Workload

Devido à carência de *workloads* para grades computacionais P2P contendo informações a respeito dos dados envolvidos no processamento das aplicações BoT, as *workloads* adotadas para as simulações realizadas neste trabalho são sintéticas e foram geradas a partir de um modelo matemático descrito detalhadamente em [Silva et al. 2009].

O gerador sintético de *workloads* permite a variação dos seguintes parâmetros de entrada: *(i)* distribuição do volume de dados, *(ii)* grau de reuso *intra-job* e *(iii)* e grau de reuso *inter-jobs*. A distribuição do volume de dados por nó é realizada segundo um fator de distribuição de carga (α) e um fator de balanceamento de carga (β). O fator de balanceamento define a quantidade de nós $N \cdot \beta$ que serão responsáveis pela fração α da carga total. Os demais nós $N \cdot (1 - \beta)$ serão responsáveis pelo restante da carga total. A Figura 1 apresenta um exemplo dessa distribuição segundo um fator de distribuição de carga $\alpha = 30\%$ e um fator de balanceamento de carga $\beta = 60\%$ para 5 nós.

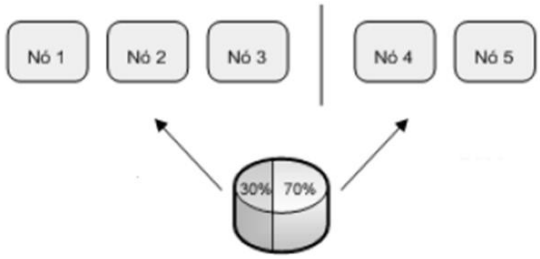


Figura 1. Balanceamento de carga entre os nós

Com estes parâmetros, é possível gerar *workloads* onde a carga gerada está bem distribuída entre todos os nós da grade, bem como *workloads* em que uma pequena porção de nós é responsável por uma grande fração da carga total. Os graus de reuso *intra-job* e *inter-jobs* definem a quantidade de dados que são reusados em um mesmo conjunto de tarefas ou em distintos conjuntos de tarefas, respectivamente.

Com este modelo foi possível gerar sinteticamente um conjunto de *workloads* com características amplamente diferenciadas entre si, que em nossa opinião contempla uma parcela representativa de *workloads* para aplicações BoT *data-intensive*.

3.2 Cenários

As simulações realizadas consideram dois cenários distintos, sendo o primeiro caracterizado pela presença de *free-riders* na grade e o segundo pela sua ausência. O primeiro cenário foi usado para demonstrar a eficiência do mecanismo ao marginalizar *free-riders* em momentos de alta contenção do espaço em disco. O segundo cenário possibilitou avaliar a justiça e a eficiência do sistema de *cache* em disco ao alocar os recursos entre os diferentes nós que compõem a grade. Em todos os experimentos de simulação, a quantidade fixa de nós considerada foi de 100 nós.

O primeiro cenário é composto por um conjunto de simulações que diferem em suas *workloads* pelo fator de balanceamento de carga (β), sendo a carga distribuída simetricamente entre os fragmentos ($\propto = 50\%$). As simulações foram divididas em dois grupos. No primeiro, a quantidade de *free-riders* (*f*) é fixada em 50% da totalidade de nós, sendo variado o fator de balanceamento de carga entre os seguintes valores: 25%, 50% e 75%. No segundo grupo a quantidade de *free-riders* assume as proporções de 25%, 50% e 75% dos nós envolvidos na grade, enquanto que o ponto de fragmentação assume o valor constante de $\beta = 50\%$. Neste cenário, todas as simulações foram realizadas em estado de alta e baixa contenção com o intuito de se avaliar a eficiência do mecanismo na presença de *free-riders*. Nos cenários de alta contenção as *caches* possuem um total de 300 G*bytes* de espaço em disco disponível para armazenamento, que representa em torno de 10% do volume total de dados. Já em cenários de baixa contenção, as *caches* possuem 2,1 T*bytes*, que representam aproximadamente 70% do volume total de dados.

O segundo cenário é composto pelo conjunto de simulações que contemplam um número bem maior de *workloads* geradas sinteticamente. As *workloads geradas* foram divididas em 3 grupos caracterizados pelo fator de balanceamento de carga (β), que assumem os seguintes valores: 25%, 50% e 75%. Para que fosse possível avaliar o uso justo e eficiente do sistema de *cache* em disco, a satisfação computada diferencia dois subgrupos, sendo o primeiro subgrupo composto por $N \cdot \beta$ nós e o segundo pelos outros

$N \cdot (1 - \beta)$ nós. De maneira análoga ao cenário anterior, a carga foi distribuída simetricamente para cada fragmento ($\propto = 50\%$). Neste cenário, todos os nós possuem o papel de colaboradores. Neste cenário foram consideradas situações de alta e média contenção, onde média contenção implica em um espaço em disco disponível para *caching* de 1,3T*bytes*, representando cerca de 40% do volume total de dados.

Para o primeiro cenário foram realizados em torno de 300 experimentos de simulação, e para o segundo foram realizados aproximadamente 4.000 experimentos de simulação, que foram suficientes para a obtenção de resultados com erro inferior a 5% para mais ou para menos, com nível de confiança de 95%.

3.3 Resultados e Análise

Nesta subseção apresentamos os resultados obtidos para a avaliação do mecanismo de *cache* em disco para grades P2P realizado através de experimentos de simulação para os dois cenários descritos anteriormente. Os gráficos apresentam em seu eixo vertical o nível de satisfação médio (vide Equação 4) entre os nós no decorrer dos turnos (eixo horizontal).

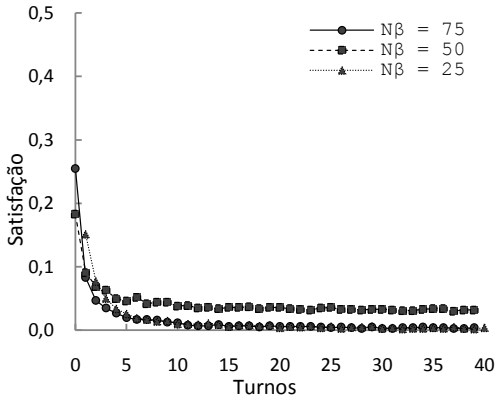
A Figura 2 apresenta os resultados para o primeiro cenário, sendo os dois primeiros gráficos (Figura 2(a) e Figura 2(b)) relacionados à execução em estado de alta contenção e os seguintes (Figura 2(c) e Figura 2(d)) ao estado de baixa contenção. Observa-se que em todos os experimentos de simulação realizados para este cenário, o mecanismo proposto para gerir a *cache* em disco em grades computacionais P2P identifica e marginaliza os *free-riders* em poucos turnos independente do nível de contenção, da quantidade de *free-riders* e do ponto de fragmentação dos nós.

Em momento de alta contenção (Figura 2(a) e Figura 2(b)) a marginalização dos *free-riders* ocorre quase que imediatamente, devido à alta demanda dos colaboradores por espaço em disco. Na Figura 2(a), para $N \cdot \beta = 75$, os *free-riders* atingem o maior nível de satisfação quando as simulações são iniciadas e as *caches* se encontram vazias. Entretanto, à medida que as *caches* entram em estado de contenção, o mecanismo privilegia os nós que possuem maiores reputações, o que, por conseqüência, leva à marginalização dos *free-riders* que têm reputação igual a zero. Devido à elevada quantidade de dados requeridos para processamento (em torno de 66% da carga total), esse cenário apresenta um nível de satisfação para os *free-riders* praticamente desprezível. Para $N \cdot \beta = 25$, os *free-riders* são responsáveis por aproximadamente 34% da carga total. Como todo o restante da carga é submetido pelos colaboradores, os espaços disponíveis nas *caches* são alocados para armazenamento de dados provenientes destes. A falta de espaço ocioso conduz o mecanismo à marginalização de nós que não contribuem com a grade. Para $N \cdot \beta = 50$, o nível de satisfação apresenta o menor índice de marginalização, que, ainda assim é satisfatório. Isto ocorre devido ao equilíbrio na distribuição da carga entre os nós.

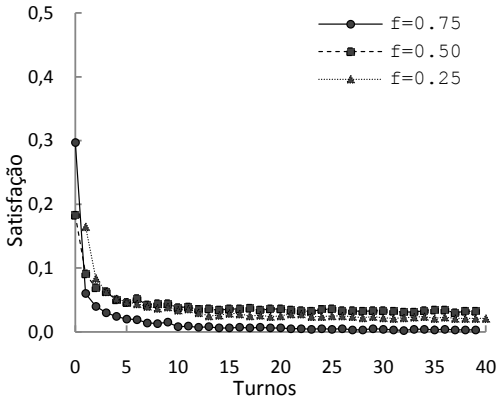
De maneira análoga, o resultado dos experimentos de simulação realizados com variação da quantidade de *free-riders* presentes na grade, apresentado na Figura 2(b), demonstra que a presença de qualquer margem de nós não-colaboradores para o sistema é rapidamente detectada, sendo estes marginalizados.

O resultado das simulações realizadas para o estado de baixa contenção (Figura 2(c) e Figura 2(d)), demonstram que na disponibilidade de espaço em disco ocioso nas *caches*, este é doado para armazenamento dos dados provenientes das tarefas

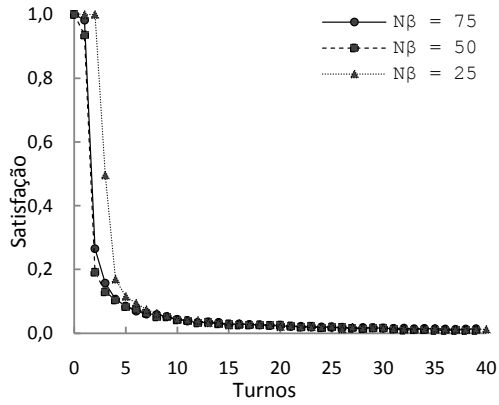
submetidas pelos *free-riders*. Contudo, ao longo dos turnos, o mecanismo prioriza o armazenamento de dados remotos pertencentes a nós que possuem reputação positiva, que por sua vez, possuem cotas, descartando os dados previamente persistidos para execução dos *free-riders*. Desta forma, temos evidências fortes de que os *free-riders* só poderão tirar proveito da grade em situações em que a oferta de espaço em disco do sistema de *cache* seja superior à demanda dos nós, ou seja, quando houver espaço ocioso.



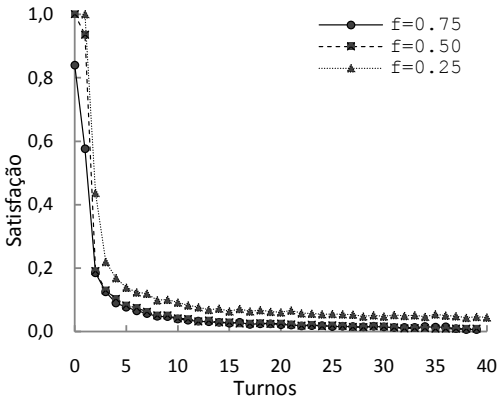
(a) Alta contenção com f = 50% e diferentes valores de $N\beta$



(b) Alta contenção com $N\beta$ = 50 e diferentes valores de f



(c) Baixa contenção com f =50% e diferentes valores de $N\beta$



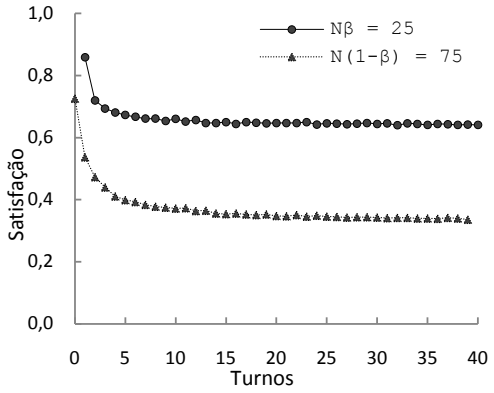
(d) Baixa contenção com $N\beta$ = 50 e diferentes valores de f

Figura 2. Nível de satisfação médio para *free-riders*

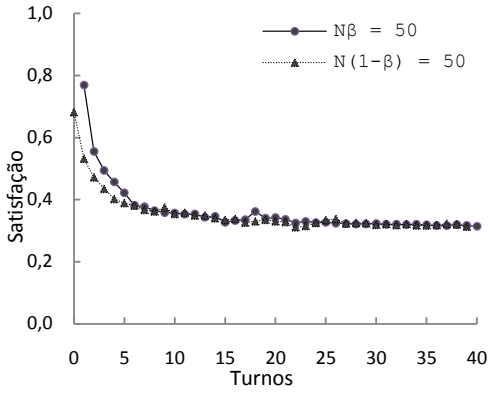
A Figura 3 mostra os resultados de satisfação dos nós colaboradores calculados a partir do segundo cenário experimental em que não há *free-riders* e o nível de contenção é médio. Neste cenário, a porção de nós que requisitam a menor quantidade de espaço em disco têm suas requisições satisfeitas em praticamente todos os casos.

Na Figura 3(a), 25 nós geram em torno de 30% da carga do sistema. Desta forma, esse pequeno conjunto de nós doa mais espaço do que necessita, permitindo que parte de seu disco seja usado para armazenar dados de nós remotos. Na Figura 3(b), todos os nós possuem a mesma carga de dados para execução na grade. Desse modo, o grau de satisfação entre os conjuntos de nós são equivalentes ao longo dos turnos, o que

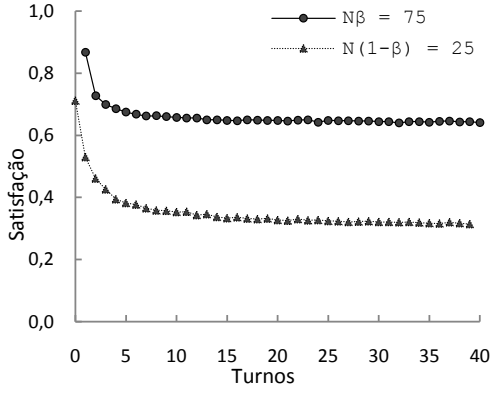
demonstra a justiça na alocação do espaço de armazenamento da *cache* entre os nós. Na Figura 3(c), 25 nós são responsáveis por 70% da carga submetida para execução na grade. Os outros 75 nós são responsáveis pela submissão de 30% da carga. A quantidade de espaço em disco ocioso é maior neste cenário do que no primeiro cenário, em que 25 nós geravam 30% da carga, pois temos aqui uma grande quantidade de nós que apresentam menor demanda do que oferta local, isto é, uma quantidade maior de nós tem sobra em disco para ser doada. Devido a esta baixa ocupação por dados locais nos 75% dos nós que compõem o sistema, a pequena parcela de nós que concentram a carga alcançam um nível de satisfação superior à mesma proporção apresentada na Figura 3(a). Este resultado demonstra o uso eficiente do espaço em disco. Observou-se também que em momentos de baixa contenção os nós com espaço ocioso chegaram a oferecer favores a nós com reputação nula, o que já era esperado haja vista nosso modelo dividir espaço ocioso igualitariamente entre nós que possuem reputação nula.



(a) Colaboradores fragmentados em $N\beta = 25$



(b) Colaboradores fragmentados em $N\beta = 50$



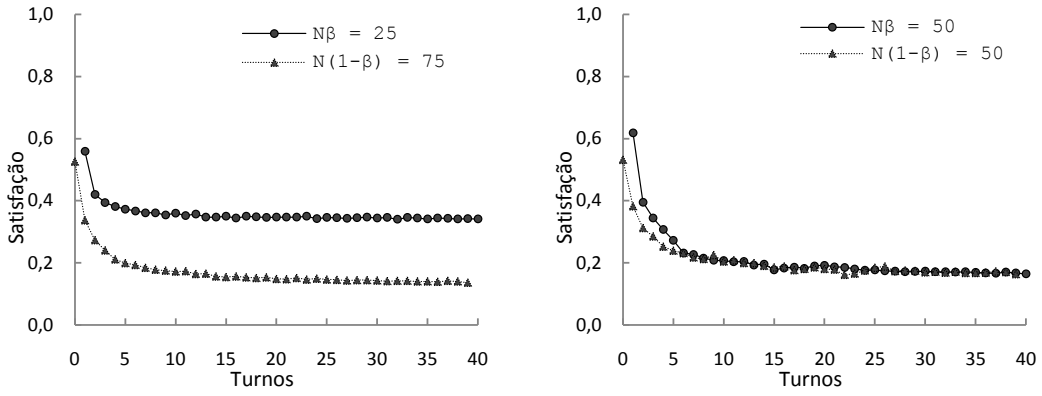
(c) Colaboradores fragmentados em $N\beta = 75$

Figura 3. Nível de satisfação médio para colaboradores (média contenção)

De maneira análoga a análise anterior, a Figura 4 apresenta os resultados de satisfação dos nós colaboradores para o segundo cenário experimental em alta contenção. Como esperado, os resultados apresentam o mesmo comportamento do

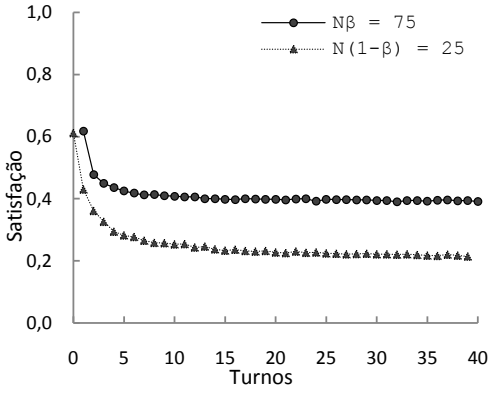
experimento em média contenção, diferindo apenas no grau de satisfação dos colaboradores.

As simulações realizadas demonstraram que o mecanismo do sistema de *cache* para grades computacionais P2P proposto neste trabalho, provê alocação justa e eficiente do espaço em disco compartilhado pelos nós, marginalizando *free-riders* em momentos de contenção de recurso.



(a) Colaboradores fragmentados em $N\beta = 25$

(b) Colaboradores fragmentados em $N\beta = 50$



(c) Colaboradores fragmentados em $N\beta = 75$

Figura 4. Nível de satisfação médio para colaboradores (alta contenção)

4. Trabalhos relacionados

A grande contribuição deste trabalho é definir um mecanismo de *caching* para grades computacionais P2P que resulte em uma distribuição justa e eficiente da capacidade de armazenamento que compõe a *cache*. Este mecanismo define o modelo de compartilhamento a ser seguido pelo sistema. De forma resumida, mecanismos de compartilhamento em sistemas P2P visam incentivar a colaboração e/ou marginalizar os *free-riders*. Em um estudo recente, Karakaya et al. classificam os métodos de incentivo à colaboração e combate a *free-riding* baseados em três abordagens: monetária, reciprocidade e reputação [Karakaya et al. 2009]. A primeira consiste em micro-

pagamentos pelos serviços providos aos nós. As soluções apresentadas para essa abordagem são caracterizadas por uma entidade centralizada responsável pelo gerenciamento dos balanceamentos e transações de cada nó. Na abordagem baseada na reciprocidade, as contribuições de um nó ao sistema ou a outros nós estabelecem o nível de contribuição do nó em uma sessão, ou seja, nós que em uma sessão são considerados *free-riders* podem ser considerados colaboradores na próxima sessão. Na terceira abordagem, baseada na reputação, a qualidade do serviço oferecido a um nó está diretamente associada à reputação deste. A reputação de um nó corresponde ao comportamento do mesmo desde sua entrada no ambiente até o presente momento. A abordagem seguida neste trabalho é baseada em reputação e por esta razão apenas políticas de compartilhamento baseadas em reputação são apresentadas a seguir.

Ngan et al. [Ngan et al. 2003] propuseram um mecanismo de alocação de espaço que limita o consumo de qualquer nó ao espaço doado pelo mesmo para a comunidade, sendo regido por uma entidade centralizada que monitora todas as transações. Esse tipo de mecanismo pode subutilizar os recursos, uma vez que espaços que não estão sendo utilizados por um nó poderão não ser aproveitados por outros nós da comunidade.

Um mecanismo descentralizado de incentivo à colaboração baseado em contratos entre os nós foi proposto por Ma e Wang [Ma & Wang 2005]. Um nó que deseja armazenar um dado estabelece um contrato com os nós com os quais deseja interagir. O espaço em disco alocado para cada nó é limitado por cotas, e caso um nó exceda sua cota, ele será punido. O mecanismo de incentivo proposto neste trabalho também define cotas, no entanto, o espaço em disco disponível para armazenamento pode ser alocado para qualquer nó enquanto não houver contenção de recursos, mesmo que os nós a serem beneficiados não tenham cota suficiente para receber os favores.

A política de *caching* proposta neste trabalho baseia-se em um mecanismo descentralizado de incentivo à colaboração e marginalização de *free-riders* chamado *rede de favores* (NoF, do inglês *Network of Favors*) [Andrade et al. 2007]. Cada nó que compõe o sistema mantém o histórico de todas as interações com outros nós do sistema para os quais aquele nó já prestou ou recebeu favores. Com base nesse histórico é possível que cada nó associe a cada nó conhecido uma reputação, que dita as interações futuras entre os nós. Quanto mais favores um nó fizer aos demais, mais favores este nó tenderá a receber dos outros nós. Este mecanismo de incentivo prioriza nós colaboradores enquanto, marginaliza os *free-riders*. Apesar da NoF ter sido proposta para o compartilhamento de recursos de processamento, sua aplicabilidade é completamente extensível ao compartilhamento de recursos de armazenamento, como foi feito neste trabalho.

O *Solomon* [Soares et al. 2008] é um mecanismo de incentivo à colaboração para sistemas P2P de armazenamento que também se baseia na rede de favores proposta por Andrade et al. No *Solomon*, o espaço compartilhado de um nó é dividido entre os nós de sua comunidade de acordo com suas reputações. Os requisitos do sistema de *backup* no qual o *Solomon* é usado são bem diferentes dos requisitos do mecanismo de *cache* propostos neste trabalho. Por conta disso, existem detalhes de projeto distintos entre o Solomon e o mecanismo aqui proposto. Uma dessas diferenças é a definição de um favor. Enquanto no *Solomon* um favor é o ato de armazenar um dado, no mecanismo aqui proposto um favor é o ato de prover um *byte* previamente armazenado. Cada vez que o dado é usado mais um favor é computado. Outra diferença é a função de avaliação

da reputação. O *Solomon* considera, além dos favores trocados entre os nós, a disponibilidade dos nós. Finalmente, uma diferença significativa deste trabalho em relação ao *Solomon* é que não necessitamos de um mecanismo explícito de auditoria para avaliação da integridade de um dado, uma vez que um favor só é contabilizado quando os dados são utilizados.

5. Conclusão e Trabalhos Futuros

Grades computacionais P2P apresentam-se como uma solução barata e eficiente para a execução de aplicações do tipo BoT. Dentre essas aplicações, podemos destacar um subconjunto que são caracterizadas pelo consumo e/ou geração de uma grande quantidade de dados, denominadas aplicações BoT *data-intensive*. A movimentação dessa grande massa de dados pode impactar diretamente no desempenho desse tipo de aplicação em grades P2P, devido à alta latência comum na comunicação entre-pares de um sistema P2P. Felizmente, constatou-se que muitas dessas aplicações apresentam um alto percentual de reuso de dados, de forma que o uso de uma estratégia de *cache* pode minimizar as perdas de desempenho relacionadas com a movimentação dos dados. Os principais requisitos deste sistema de *cache* em disco para grades computacionais P2P são: *(i)* alocação justa e eficiente do espaço em disco disponível para a *cache* e *(ii)* incentivo à colaboração.

Neste trabalho apresentamos um mecanismo de gerência de *cache* em disco que incentiva a doação de recursos ao sistema e resulta em uma alocação justa e eficiente do espaço em disco disponível para cada um dos nós participantes da grade P2P. Tal mecanismo é descentralizado e não necessita de auditorias. O mecanismo privilegia nós colaboradores em detrimento dos nós que não contribuem com o sistema (*free-riders*). Quando não há contenção de recursos, o mecanismo permite que qualquer nó, mesmo que não seja um colaborador, possa usufruir do espaço ocioso disponível para armazenamento, levando à maximização da utilização dos recursos. Em estado de alta contenção, a alocação do espaço disponível será baseada em um mecanismo de reputação inspirado na rede de favores.

A avaliação do mecanismo, realizada através de experimentos de simulação, demonstra que o mecanismo de *cache* em disco proposto neste trabalho viabiliza o suporte a aplicações BoT *data-intensive*, provendo eficiência e justiça para o sistema de *cache* em disco para grades computacionais entre-pares. Os experimentos mostraram que em estado de alta contenção, *free-riders* são rapidamente detectados e marginalizados, enquanto que em estado de baixa contenção, eles conseguem tirar proveito do espaço ocioso em disco uma vez que tenta-se maximizar o uso dos recursos compartilhados. Demonstramos, também, que na ausência de *free-riders*, o mecanismo realiza a alocação do espaço de forma bastante justa, proporcional à contribuição de cada nó ao sistema.

Dentre os trabalhos futuros, incluímos o estudo da redução do tempo de execução das aplicações *data-intensive* que executam na grade P2P onde o serviço de *caching* é provido. Estudos preliminares demonstram que a *cache* evita a transferência de 40% do total de dados que deveria ser transferido caso o serviço de *cache* não existisse. Outros direcionamentos para trabalhos futuros envolvem o estudo da escalabilidade do nosso mecanismo para grades P2P em cenários compostos por 1.000 e

10.000 nós, além de avaliar o desempenho do mecanismo através da realização de experimentos de medição em uma grade P2P em produção, como o *OurGrid*[2].

Referências

Adar, E. and Huberman, B. A. (2000) "Free riding on gnutella", First Monday, 5(10).

Andrade, N., Brasileiro, F., Cirne, W. and Mowbray, M. (2007) "Automatic Grid Assembly by Promoting Collaboration in Peer-to-Peer Grids", In: Journal of Parallel and Distributed Computing, v. 67, n. 8, p957-966.

Foster, I. Kesselman, C. Nick, J. M. and Tuecke, S. (2002) "The Physiology of the Grid: An Open Grid Service Architecture for Distributed Systems Integration", Open Service Infrastructure WG, Global Grid Forum.

Karakaya, M., Korpeoglu, I. and Ulusoy, Ö. (2009) "Free Riding in Peer-to-Peer Networks", In: Internet Computing, IEEE, Vol. 13, No. 2, p92-98.

Ma, Y. and Wang, D. (2005) "An Approach to Fair Resource Sharing in Peer-to-Peer Systems", In: Networking – ICN 2005, Vol. 3421/2005, p643-652.

Ngan, T., Wallach, D. and Druschel, P. (2003) "Enforcing Fair Sharing of Peer-to-Peer Resources", In: Proceedings of the 2nd International Workshop on Peer-to-Peer Systems.

Parada, J., Perdomo, E., Rivas, R. and Ruette, F. (2009) "Cativic: A Grid Experience on Quantum Chemistry", In: Proceedings of the First EELA-2 Conference, Colombia, Editorial CIEMAT, p477-486.

Podlipnig, S. and Böszörmenyl, L. (2003) "A Survey of Web Cache Replacement Strategies", In: ACM Computing Surveys, Vol. 35, No. 4, p374-398.

Santos-Neto, E., Cirne, W., Brasileiro, F. and Lima, A. (2004) "Exploiting Replication and Data Reuse to Efficiently Schedule Data-Intensive Applications on Grids", In: Job Scheduling Strategies for Parallel Processing, Vol. 3277/2004, p210-232.

Silva, F., Gagliard, H., Gallo, E., Madope, M., Neto, V., Pisa, I. and Alves, D. (2007) "IntegraEPI: a Grid-based Epidemic Surveillance System", In: Proceedings of the HealthGrid Conferecen, Switzerland.

Silva, R., Brasileiro, F. and Lopes, R. (2009) "Um Serviço de Cache para Grades Computacionais Entre-Pares", Relatório Técnico, Universidade Federal de Campina Grande, Brasil. Disponível em: http://www.lsd.ufcg.edu.br/~rafael/tech_report.pdf.

Soares, P., Oliveira, M., Guerrero, D. and Brasileiro, F. (2008) "Solomon: Incentivando o Compartilhamento e Maior Disponibilidade em Sistemas de Armazenamento Entre-Pares", In: Anais do IV Workshop on Peer-to-Peer, Sociedade Brasileira de Computação, v. 1, p1-12.

Verleyen, J., Hurtado-Ramirez, J. M. and Merino-Perez, E. (2009) "Meta-Dock, a gridification of a docking application", In: Proceedings of the First EELA-2 Conference, Colombia, Editorial CIEMAT, p147-154.

[2] http://www.ourgrid.org