Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Informática

Dissertação de Mestrado

Um Serviço de *Cache* de Disco para Grades Computacionais Entre-Pares

Rafael Ferreira da Silva

Campina Grande, Paraíba, Brasil
Fevereiro - 2010

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Informática

Um Serviço de *Cache* de Disco para Grades Computacionais Entre-Pares

Rafael Ferreira da Silva

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

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos

Francisco Vilar Brasileiro
(Orientador)
Raquel Vigolvino Lopes
(Orientadora)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S586s
2010 Silva, Rafael Ferreira da.
Um serviço de *cache* de disco para grades computacionais entre-pares / Rafael Ferreira da Silva. — Campina Grande, 2010.
73 f. : il.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Referências.
Orientadores: Prof. Dr. Francisco Vilar Brasileiro, Profª. Drª. Raquel Vigolvino Lopes.

1. Redes de Computadores. 2. Grades Computacionais. 3. Sistemas P2P. I. Título.

CDU – 004.7(043)

"UM SERVIÇO DE CACHE DE DISCO PARA GRADES COMPUTACIONAIS ENTRE-PARES"

RAFAEL FERREIRA DA SILVA

DISSERTAÇÃO APROVADA EM 26.02.2010

FRANCISCO VILAR BRASILEIRO, Ph.D
Orientador(a)

RAQUEL VIGOLVINO LOPES, D.Sc
Orientador(a)

CARLOS EDUARDO SANTOS PIRES, Dr.
Examinador(a)

DENIO MARIZ TIMÓTEO DE SOUSA, Dr.
Examinador(a)

CAMPINA GRANDE - PB

.

"Se eu pudesse dar só uma dica sobre o futuro seria esta: use filtro solar."

Mary Schmich.

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 à políticas de utilização não necessariamente homogêneas, não é uma tarefa trivial. Neste trabalho nós propomos um mecanismo de gerência de *cache* de disco que viabiliza o suporte a este tipo de aplicação através da redução da quantidade de transferências entre domínios administrativos. O mecanismo 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. Através de experimentos de simulação, demonstrou-se que o mecanismo é capaz de reduzir a quantidade de transferência de dados entre domínios administrativos e que, em estado de contenção de recursos, nós não-colaboradores (*free-riders*) são marginalizados em favor dos nós que colaboram com a grade, independentemente da quantidade de nós que compõem o sistema. Por fim, um estudo de caso realizado em uma grade computacional P2P controlada mostrou que o mecanismo é capaz de reduzir a quantidade de transferências que, por conseqüência, reduz o tempo de execução das aplicações, quando comparado à execução sem a utilização do mecanismo de gerência de cache de disco proposto.

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 due to data transfer does not impact application performance. Fortunately, many of these applications have a high rate of data reuse, 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 work we propose a mechanism for disk cache management that gives support to this type of application by reducing the number of transfers amongst administrative domains. The mechanism 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. Through simulation experiments, we demonstrated that the mechanism has reduced the amount of data transfers amongst administrative domains and during resource contention free-riders are marginalized in favor of collaborators, regardless of the number of nodes that comprise the system. Finally, a case study executed over a controlled P2P computational grid showed that the mechanism has reduced the amount of transfers, which consequently reduced the execution times of the applications when compared to the execution of the same applications without using the disk cache management mechanism proposed.

Agradecimentos

Primeiramente, agradeço a Deus pelo dom da vida e por ter me dado sabedoria para conduzir este trabalho.

Agradeço, de forma especial, ao meu grande amor, minha noiva Stéfani, que além de ser companheira e amiga, me deu todo o apoio necessário para a conclusão deste trabalho, especialmente em momentos conturbados. Obrigado pela paciência e carinho dedicado a mim.

Agradeço aos meus pais, Wellington e Célia, e aos meus irmãos, Danielly e Sandro, que, apesar da distância, sempre se fizeram presentes apoiando minhas decisões.

Agradeço aos meus orientadores Prof. Francisco Brasileiro (Fubica) e Profa. Raquel Lopes pela orientação constante e precisa, e pelo incentivo e confiança depositada em mim.

Aos amigos do LSD, em particular aos integrantes da sala Volúpia que, em momentos de feriados e fim de semana estavam sempre reunidos no laboratório. Agradeço em especial ao meu amigo Lesandro pelos momentos de discussões filosóficas e estatísticas.

Agradeço ao Prof. Giuseppe Mongiovi (*in memorian*) por ter me ensinado os primeiros passos para o desenvolvimento da pesquisa, e que, além de professor, foi um amigo.

Obrigado!

Conteúdo

1 Introdução **1**

1.1 Estado da Arte 4

1.2 Objetivos do Trabalho 5

1.3 Escopo do Trabalho 6

1.4 Organização da Dissertação 6

2 *Cache* de Disco em Grades Computacionais P2P **7**

2.1 Definição do Problema 7

2.2 Serviço de *Cache* de Disco em Grades P2P 9

2.3 Mecanismo de *Caching* Baseado em Reputação 11

2.4 Considerações Finais 15

3 Avaliação Experimental **17**

3.1 Modelagem da *Workload* 18

3.2 Avaliação do Impacto do Uso de *Cache* em Disco 21

3.2.1 Descrição dos Cenários 21

3.2.2 Resultados e Análise 23

3.3 Eficiência e Justiça 26

3.3.1 Descrição dos Cenários 26

3.3.2 Resultados e Análise 29

3.4 Escalabilidade 34

3.4.1 Descrição dos Cenários 34

3.4.2 Resultados e Análise 35

3.5 Considerações Finais 38

4 Estudo de Caso **39**

4.1 OurGrid 39

4.2 Implementação do Protótipo do Serviço de *Cache* 41

4.3 Experimentação 42

4.3.1 Descrição da Aplicação 44

4.3.2 Descrição dos Cenários 44

4.3.3 Resultados e Análise 46

4.4 Considerações Finais 48

5 Trabalhos Relacionados **49**

5.1 Sistemas P2P de Armazenamento *versus Caching* em Grades P2P 49

5.2 Mecanismos de Incentivo à Colaboração 50

5.3 Considerações Finais 52

6 Conclusões e Trabalhos Futuros **53**

A Resultados dos Experimentos **59**

A.1 Avaliação do Impacto do Uso de *Cache* em Disco 59

A.2 Eficiência e Justiça 61

A.3 Escalabilidade 69

A.4 Estudo de Caso 73

Lista de Acrônimos

BoT - *Saco-de-Tarefas (do inglês Bag-of-Tasks)*

GDSF - *Greedy-Dual Size with Frequency*

LFU - *Least Frequently Used*

LFU-DA - *LFU with Dynamic Aging*

LRU - *Least Recently Used*

NFS - *Sistema de Arquivos em Rede (do inglês Network File System)*

NoF - *Rede de Favores (do inglês Network-of-Favors)*

P2P - *Entre-Pares (do inglês Peer-to-Peer)*

XMPP - *eXtensible Messaging and Presence Protocol*

Lista de Figuras

1.1 Arquitetura de uma grade computacional P2P 2

2.1 Armazenamento de um dado local com remoção de dados expirados e remotos 10
2.2 Utilização de cotas elásticas para divisão do espaço entre nós remotos . . . 11

3.1 Balanceamento de carga entre os nós 19
3.2 Taxa de acertos da *cache* em M*Bytes* para diferentes valores de reuso *intra-job* (r_{intra}) 24
3.3 Taxa de acertos da *cache* em M*Bytes* para diferentes valores de reuso *inter-jobs* (r_{inter}) 25
3.4 Taxa de acertos da *cache* em M*Bytes* para duas configurações específicas . . 27
3.5 Nível de satisfação médio para *free-riders* 30
3.6 Nível de satisfação médio para colaboradores (média contenção) 32
3.7 Nível de satisfação médio para colaboradores (alta contenção) 33
3.8 Nível de satisfação médio para colaboradores (c) e *free-riders* (f) para 1.000 nós 36
3.9 Nível de satisfação médio para colaboradores (c) e *free-riders* (f) para 10.000 nós 37

4.1 Arquitetura do *OurGrid* 40
4.2 Formas de realizar *caching* no *OurGrid* 41
4.3 Gerenciamento de requisições para a *cache* compartilhada 43
4.4 Comparação entre estratégias de armazenamento 47
4.5 Medição do *makespan* (a) e do nível de satisfação (b) entre colaborador e *free-rider* 48

Lista de Tabelas

3.1 Características do cenário para avaliação do impacto do uso de *cache* em disco 23

3.2 Características do primeiro cenário para avaliação da alocação justa e eficiente do sistema de *cache* 28

3.3 Características do segundo cenário para avaliação da alocação justa e eficiente do sistema de *cache* 29

3.4 Características dos cenários para avaliação da escalabilidade 35

4.1 Dados de entrada para o primeiro cenário 45

4.2 Dados de entrada para o segundo cenário 46

A.1 Taxa de acertos da *cache* em M*Bytes* para $r_{intra} = 25\%$ 59

A.2 Taxa de acertos da *cache* em M*Bytes* para $r_{intra} = 50\%$ 59

A.3 Taxa de acertos da *cache* em M*Bytes* para $r_{intra} = 75\%$ 59

A.4 Taxa de acertos da *cache* em M*Bytes* para $r_{inter} = 25\%$ 60

A.5 Taxa de acertos da *cache* em M*Bytes* para $r_{inter} = 50\%$ 60

A.6 Taxa de acertos da *cache* em M*Bytes* para $r_{inter} = 75\%$ 60

A.7 Taxa de acertos da *cache* em M*Bytes* para $r_{intra} = 75\%$ e $r_{inter} = 25\%$ . . . 60

A.8 Taxa de acertos da *cache* em M*Bytes* para $r_{intra} = 25\%$ e $r_{inter} = 75\%$ . . . 60

A.9 Nível de satisfação médio para *free-riders* - alta contenção com $f = 50\%$ e diferentes valores de $N \cdot \beta$ 61

A.10 Nível de satisfação médio para *free-riders* - alta contenção com $N \cdot \beta = 50$ e diferentes valores de f 62

A.11 Nível de satisfação médio para *free-riders* - baixa contenção com $f = 50\%$ e diferentes valores de $N \cdot \beta$ 63

A.12 Nível de satisfação médio para *free-riders* - baixa contenção com $n = 50$ e diferentes valores de f 64
A.13 Nível de satisfação médio para colaboradores (média contenção), $N \cdot \beta = 25$ e $N \cdot \beta = 50$ 65
A.14 Nível de satisfação médio para colaboradores (média contenção), $N \cdot \beta = 75$ 66
A.15 Nível de satisfação médio para colaboradores (alta contenção), $N \cdot \beta = 25$ e $N \cdot \beta = 50$ 67
A.16 Nível de satisfação médio para colaboradores (alta contenção), $N \cdot \beta = 75$. 68
A.17 Nível de satisfação médio para colaboradores (c) e *free-riders* (f) para 1.000 nós, $c = 25\%$ e $c = 50\%$ 69
A.18 Nível de satisfação médio para colaboradores (c) e *free-riders* (f) para 1.000 nós, $c = 75\%$ 70
A.19 Nível de satisfação médio para colaboradores (c) e *free-riders* (f) para 10.000 nós, $c = 25\%$ e $c = 50\%$ 71
A.20 Nível de satisfação médio para colaboradores (c) e *free-riders* (f) para 10.000 nós, $c = 75\%$ 72
A.21 Comparação entre estratégias de armazenamento 73
A.22 Medição do *makespan* entre colaborador e *free-rider* 73
A.23 Medição do nível de satisfação entre colaborador e *free-rider* 73

Capítulo 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 [12]. 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 recursos computacionais ociosos. Neste trabalho, consideramos uma grade computacional em que cada domínio administrativo (*site*) 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. A Figura 1.1 mostra a arquitetura de uma grade computacional P2P com múltiplos domínios administrativos, onde cada domínio possui um *peer* que gerencia seus recursos. Quando uma tarefa submetida pelo usuário é executada por um recurso em um mesmo domínio administrativo, diz-se que a tarefa foi submetida localmente. Em contrapartida, quando a execução é realizada por um recurso pertencente a outro domínio, diz-se que a tarefa foi submetida remotamente.

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 inde-

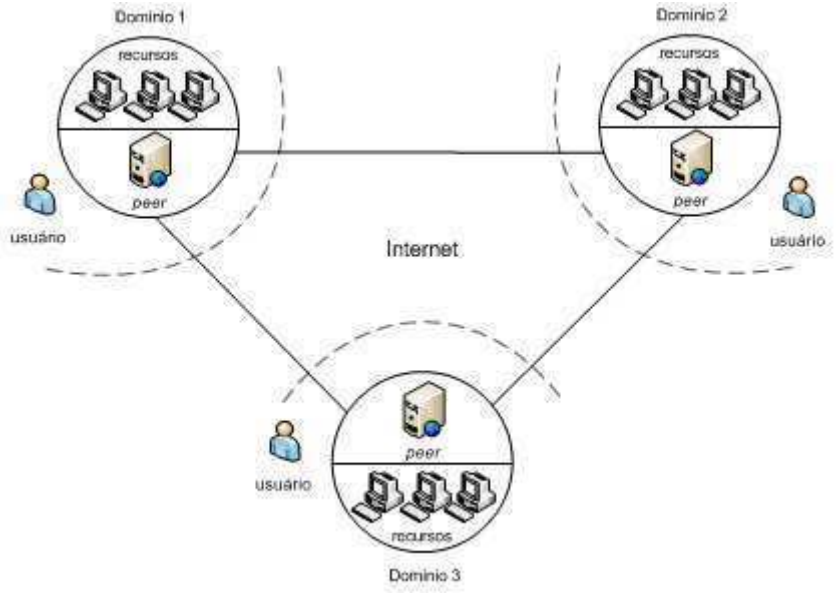


Figura 1.1: Arquitetura de uma grade computacional P2P

pendentemente 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)* a largura de banda entre os recursos que compõem a grade e *(ii)* o espaço em disco nestes recursos para o armazenamento simultâneo dos dados de todas as aplicações sendo executadas.

Os recursos 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 recursos de uma grade computacional P2P pode ser lenta e o tempo necessário para transportar dados entre recursos 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 [26; 20] e/ou em execuções sucessivas [24]. Neste

trabalho esses padrões são denominados de reuso *intra-job* e *inter-jobs*, respectivamente. Desta forma, as limitações de banda podem ser reduzidas através da persistência dos dados nos recursos de execução, evitando que dados que serão utilizados em futuras execuções sejam novamente transferidos. Nesse sentido, um sistema de *cache* de disco pode viabilizar a execução eficiente de aplicações *data-intensive* em grades P2P.

No contexto de grades computacionais P2P, o armazenamento, assim como a computação, obedece a lei do "melhor possível" (*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 pares. O problema consiste em alocar os espaços de forma justa - proporcional à contribuição dos nós; e eficiente - de forma a melhorar 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* de disco, daí a importância deste incentivo. O terceiro desafio é definir um mecanismo de gerência de *cache* de disco capaz de lidar com *free-riders*, 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 [5]. Eles reportaram que aproximadamente 70% dos nós não compartilhavam dados, e 25% destes 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* de 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

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

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. Diferentemente de outras soluções como, por exemplo, a de Soares et al. [25], 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* de disco pode ser útil apenas quando esses dados gerados servem de entrada para outras aplicações que também executarão na grade. Esse tipo de *workflow* é também comum em aplicações científicas.

1.1 Estado da Arte

Até o presente momento é desconhecida a existência de trabalhos que utilizem um sistema de *cache* de dados em grades computacionais P2P. Partindo desse pressuposto, este trabalho apresenta soluções inspiradas em mecanismos propostos para sistemas P2P de armazenamento.

Uma diferença fundamental entre sistemas P2P de armazenamento e *caching* de dados em grades P2P é quanto ao propósito de cada sistema. Em sistemas de *cache* de dados para grades P2P, busca-se melhorar o desempenho das aplicações através da redução da quantidade de movimentação de dados. Entretanto, não existem garantias de que o dado persistido estará disponível quando requisitado. Por outro lado, em sistemas P2P de armazenamento a disponibilidade de um dado é um fator crucial para a eficiência do mesmo. Essa categoria de sistema se propõe a armazenar remotamente dados de um usuário e prover mecanismos de recuperação destes. Nesse sentido, mecanismos de replicação e tolerância a falhas são serviços inerentes a sistemas P2P de armazenamento.

Desde a constatação de Adar e Huberman [5] sobre os *free-riders*, e posteriormente por Hughes et al. [14], diversos mecanismos de incentivo à colaboração e marginalização de *free-riders* têm sido propostos. Os mecanismos de incentivo propostos para sistemas P2P de armazenamento podem ser classificados em mecanismos baseados em abordagens monetárias, em reciprocidade e em reputação [15]. A primeira é definida por micropagamentos

a um nó pelos serviços providos aos demais nós. Em abordagens baseadas em reciprocidade e reputação, as contribuições de um nó estabelecem o nível de contribuição do nó em uma sessão ou desde sua entrada no sistema, respectivamente. O nível de contribuição define a qualidade do serviço ofertado a um nó.

Dentre os mecanismos propostos, diversos apresentam uma solução centralizada, onde uma entidade é responsável pelo balanceamento dos recursos e validação das transações entre os nós. O uso de entidades centralizadas acarreta dois problemas principais: *(i)* a existência de um único ponto de falha, e *(ii)* a necessidade de uma infra-estrutura de autenticação.

Outra forma de incentivo à colaboração se dá através da atribuição do espaço de armazenamento para um nó proporcional à sua contribuição ao sistema, ou seja, um nó não poderá consumir mais do que doa [17]. Esse tipo de mecanismo pode subutilizar o espaço em disco, uma vez que espaços ociosos não poderão ser utilizados por outros nós.

Uma terceira forma de incentivo é através da utilização de soluções descentralizadas baseadas em reciprocidade ou reputação de um nó [6]. Entretanto, essas soluções apresentam um novo desafio: a verificação da integridade dos dados armazenados, que pode ser realizada através da aplicação de mecanismos de auditoria [25]. O uso de mecanismos de auditoria gera um *overhead* de comunicação na rede.

1.2 Objetivos do Trabalho

O objetivo deste trabalho é propor um serviço descentralizado de *cache* de disco para grades computacionais P2P que seja capaz de viabilizar o suporte a aplicações BoT *data-intensive*, através da(o):

1. redução da transferência de dados entre domínios administrativos;
2. realização da alocação justa e eficiente do espaço em disco doado para armazenamento dos dados;
3. incentivo à colaboração entre os nós envolvidos na grade;
4. marginalização de nós que consomem mas não doam recursos à grade (*free-riders*);

1.3 Escopo do Trabalho

Este trabalho endereça a problemática relacionada às grades computacionais entre-pares, com ênfase no suporte à aplicações que apresentam um processamento intensivo de uma grande quantidade de dados (aplicações *data-intensive*) com reuso de dados, seja em uma mesma execução (*intra-job*) ou em execuções sucessivas (*inter-jobs*). O trabalho se propõe a apresentar uma solução de um sistema de *cache* de disco que reduza as limitações impostas pela grade, como também a problemática relacionada a esse tipo de sistema.

1.4 Organização da Dissertação

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2 é apresentado o serviço de *cache* de disco para grades P2P, seu funcionamento e o mecanismo de *caching* baseado em reputação para alocação de dados em momentos de contenção de recursos. Os resultados da avaliação do impacto do uso do sistema de *cache*, assim como a eficiência do mecanismo, obtidos através de experimentos de simulação, são apresentados no Capítulo 3. No Capítulo 4 é apresentado um estudo de caso realizado em uma grade computacional P2P em produção. O Capítulo 5 contextualiza este trabalho, apresentando os trabalhos relacionados, suas deficiências e diferenças em relação ao serviço apresentado neste trabalho. Finalmente, no Capítulo 6 são apresentadas as conclusões e direções para trabalhos futuros.

Capítulo 2

Cache de Disco em Grades Computacionais P2P

Neste capítulo é apresentado o serviço de *cache* de disco para grades computacionais P2P, seu funcionamento e o mecanismo baseado em reputação para alocação de dados em momentos de contenção de recursos. Na Seção 2.1 é apresentado em mais detalhes o problema a ser resolvido, como também os principais requisitos a serem atendidos pela solução. Na Seção 2.2 o serviço de *cache* de disco proposto neste trabalho é apresentado em detalhes. A Seção 2.3 complementa a especificação do serviço, sendo apresentado o mecanismo baseado em reputação para a alocação justa e eficiente do espaço em disco disponível para *caching*.

2.1 Definição do Problema

Seja $P = \{p_1, p_2, ..., p_n\}$ o conjunto de nós que compõem a grade P2P. Cada nó $p_i \in P$ representa o espaço ocioso em disco doado para armazenamento dos dados requeridos para processamento das aplicações. Nesse sentido, P representa o espaço total potencialmente disponível para armazenamento em um sistema de *cache*. Cada nó possui também um conjunto de usuários que submetem periodicamente um conjunto de tarefas BoT *data-intensive* para execução na grade. Uma tarefa k especifica uma quantidade de *bytes* a ser armazenada no sistema de *cache* de disco requerida para execução.

A cada instante de tempo t, um conjunto de tarefas $T(t) = \{k_1, k_2, ..., k_m\}$ submetido para execução na grade gera uma demanda $D(t)$ de espaço em disco para armazenamento de

dados. Seja $S_z(t)$ o espaço que o nó z disponibiliza para o sistema de *cache* em um instante de tempo t. Desta forma, o espaço total potencialmente disponível para armazenamento em um instante t é definido por:

$$S(t) = \sum_{z \in P} S_z(t) \tag{2.1}$$

Em momentos em que não há contenção, $D(t) \leq S(t)$, todas as requisições são atendidas pelo sistema de *cache*. Entretanto, em momentos de contenção, $D(t) > S(t)$, apenas uma parcela das requisições serão satisfeitas. Desta forma, a alocação do espaço em disco neste cenário deve ser realizada de forma justa e eficiente, privilegiando os nós que mais contribuíram com a grade. Uma forma de alocar o espaço é através da divisão de cotas. Cada nó $z \in P$, possui em um instante de tempo t uma cota $Q_z(t)$, de forma que:

$$\sum_{z \in P} Q_z(t) = S(t) \tag{2.2}$$

Dentre os nós envolvidos na grade, existe uma parcela $F = \{f_1, f_2, ..., f_j\} \subset P$, $j < n$, de nós que consomem mas não doam recursos à grade, denominados *free-riders*. O serviço de *cache* deve ser capaz de lidar com nós não-colaboradores, marginalizando-os em momentos de contenção de recursos e incentivando-os a colaborarem.

Para atender as proposições acima apresentadas, um serviço de *cache* de disco para grades computacionais P2P deve satisfazer os seguintes requisitos:

1. apresentar uma solução descentralizada para gerir o serviço de *caching*;
2. cada nó deve ser autônomo quanto à decisão da alocação do espaço em disco;
3. a alocação do espaço em disco deve ser proporcional à contribuição dos nós;
4. a utilização do espaço em disco deve ser maximizada, aumentando a eficiência do serviço;
5. nós não-colaboradores devem ser desencorajados a ficarem saindo e entrando na rede com novas identidades para se livrarem de más reputações (*white-washing*).

A utilização de uma solução descentralizada exclui a possibilidade de existência de um único ponto de falha para o sistema, e minimiza o *overhead* de comunicação na rede. A au-

tonomia de um nó evita que ataques provenientes de *free-riders* possam interferir na decisão da alocação do espaço em disco. Como exemplo desses ataques, pode-se citar o "ataque de conluio", onde um grupo de *free-riders* conspira para promover um ou mais nós não-colaboradores, ou prejudicar a reputação de um nó colaborador, e "ataques de adulteração", onde valores de reputação ou contribuição podem ser adulterados em sistemas que se utilizam da publicação desses valores para computar cotas, favores, etc. [15].

A divisão em cotas do espaço em disco disponível para o sistema de *cache* baseada na contribuição dos nós é uma forma de se obter justiça e incentivar a colaboração. Entretanto, o sistema deve possuir um mecanismo que seja capaz de melhorar a utilização do espaço em *cache* ocioso, isto é, a parcela da cota não utilizada pelos nós.

2.2 Serviço de *Cache* de Disco em Grades P2P

O serviço de *cache* de disco para grades computacionais P2P apresentado neste trabalho provê um sistema de armazenamento em disco descentralizado composto por nós que doam os espaços ociosos em disco de seus recursos para persistência dos dados requeridos para a execução de suas aplicações. O espaço em disco doado por um nó prioriza a persistência dos dados das aplicações que são submetidas localmente (dados locais). A *cache* de disco nunca se recusa a armazenar um dado local, exceto quando o tamanho do dado for maior que a *cache*. A cada dado local armazenado na *cache* de disco é associado um tempo de vida[1] (*lifetime*) que, uma vez expirado, marca o dado para substituição, o que provê mais espaço para armazenamento de dados das aplicações submetidas remotamente (dados remotos). A cada nova requisição de alocação de espaço para um dado local, o sistema de *cache* verifica se existe espaço livre disponível para armazenamento. Caso não exista, o sistema remove primeiramente dados locais marcados como expirados, e se necessário, o sistema remove dados pertencentes a nós remotos que estão armazenados nesse nó. Em última instância, os dados locais que ainda não expiraram serão removidos para a persistência dos novos dados. A Figura 2.1 apresenta um exemplo desse procedimento, onde o dado a ser armazenado demanda mais espaço do que o disponível para armazenamento. Como conseqüência, dados expirados e uma quantidade de dados remotos serão removidos da *cache* para ceder espaço

[1] O tempo de vida de um dado denota a quantidade máxima de tempo (ex.: segundos, horas) que um dado poderá permanecer alocado no sistema de *cache* de disco.

para armazenamento do novo dado local.

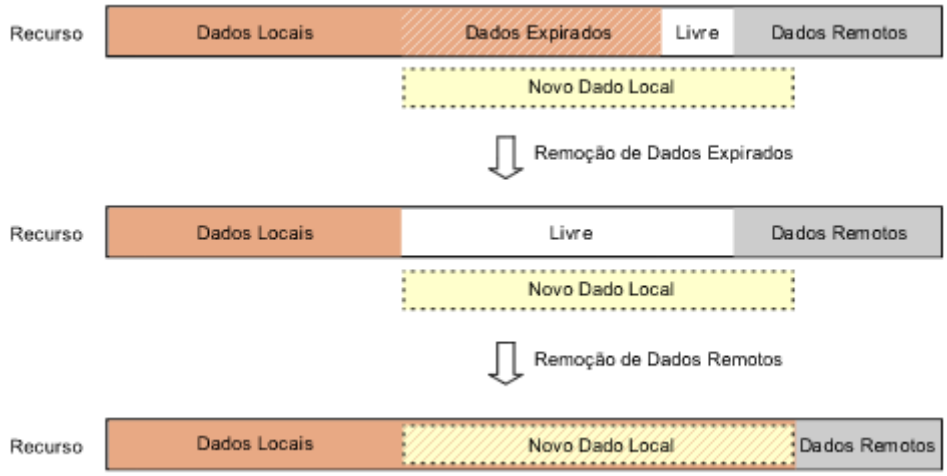


Figura 2.1: Armazenamento de um dado local com remoção de dados expirados e remotos

A definição de tempo de vida associado a um dado durante a escrita do mesmo em uma *cache* também é uma estratégia adotada para garantir que o dado será removido da *cache* em algum momento caso ocorra falha de transferência irreparável e/ou não-identificada. Sempre que um dado é utilizado, o seu *lifetime* é redefinido a partir deste momento. Essa estratégia garante que dados que são freqüentemente utilizados não terão seu tempo de vida expirado prematuramente.

Os dados remotos são armazenados nos espaços não utilizados pelos dados locais. A alocação desse espaço é regida por um mecanismo de cotas que estabelece a fração de disco do nó z em que cada nó remoto pode armazenar seus dados. As cotas estabelecidas para cada nó são elásticas, ou seja, um nó pode armazenar dados no espaço ocioso do disco remoto, mesmo que exceda sua cota. Entretanto, um nó que excede sua cota pode ter dados removidos prematuramente do sistema de cache de disco quando o espaço for requisitado por outros nós. Como exemplo, sejam p_1 e p_2 dois *nós* conhecidos de um nó z, com cotas iguais a Q_1 e Q_2, respectivamente. No primeiro momento, p_1 possui uma demanda superior a sua cota Q_1. Como a demanda de p_2 é inferior a Q_2, p_1 utiliza o espaço não utilizado por p_2 para armazenamento de seus dados. Posteriormente, quando p_2 requisita o espaço estabelecido por sua cota Q_2, dados excedentes de p_1 serão removidos. A Figura 2.2 ilustra esta situação.

A substituição de dados da *cache* pode seguir qualquer estratégia de substituição de dados, como por exemplo, LRU (*Least Recently Used*) ou LFU (*Least Frequently Used*) [21].

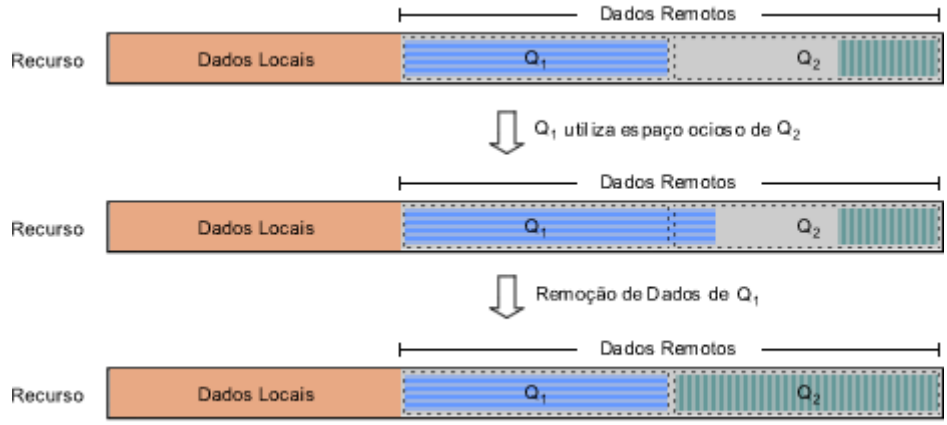


Figura 2.2: Utilização de cotas elásticas para divisão do espaço entre nós remotos

As estratégias de substituição definem em que ordem os dados serão removidos da *cache*, através da atribuição de valores aos dados. Como exemplo desses valores, pode-se citar a freqüência, data de utilização e tamanho do dado.

2.3 Mecanismo de *Caching* Baseado em Reputação

O mecanismo de *cache* de disco provê uma alocação justa e eficiente do espaço disponível para armazenamento compartilhado entre os nós de uma grade P2P. Em momentos de contenção de recursos, a alocação do espaço ocioso disponibilizado para armazenamento remoto será baseada em um mecanismo de priorização inspirado na rede de favores proposta por Andrade et al. [6]. 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 um nó dite as interações futuras entre os nós conhecidos. No contexto da *cache* de disco, um favor é definido como a ação de se prover um *byte* de dado para a execução de uma tarefa submetida por um usuário pertencente a 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 2.3.

$$B_z^y(t) = max\left[0, b_z^y(t) - b_y^z(t)\right] \tag{2.3}$$

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 (favores recebidos de y). $b_y^z(t)$ denota a quantidade de *bytes* providos pelo nó z ao nó y para processamento de tarefas até o instante t (favores ofertados a y).

Seja $K \subseteq P$, 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:

$$R_z^y(t) = \begin{cases} \frac{B_z^y(t)}{\sum_{x \in K} B_z^x(t)},\ se\ \sum_{x \in K} B_z^x(t) \neq 0 \\ 0,\ caso\ contrário \end{cases} \tag{2.4}$$

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:

$$Q_z^y(t) = R_z^y(t) \cdot S_z(t) \tag{2.5}$$

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

A cota baseada em reputação garante que a quantidade de espaço na *cache* de disco de um nó z oferecida para um nó y 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, assim, estes aumentam suas reputações 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. [25] e foi tomada de forma a melhorar o uso do espaço disponível para a *cache* de 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* de disco quando o espaço for requisitado por outros nós com 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 2.1 apresenta uma visão geral do procedimento para alocação do espaço em disco para um dado. Os Algoritmos 2.2 e 2.3 apresentam o procedimento detalhado para alocação do espaço em disco para dados locais e remotos, respectivamente. Três características destes algoritmos de armazenamento de dados 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, o processo inicia-se pelos nós com menor reputação (Algoritmo 2.2, linha 10 e Algoritmo 2.3, linha 3), existindo a chance de não ser necessário apagar dados excedentes de nós com maior reputação. Finalmente, os próprios dados do nó (mesmo que não sejam excedentes) podem ser removidos para dar lugar aos novos dados (Algoritmo 2.2, linha 5 e Algoritmo 2.3, linha 7). Esta abordagem de alocação de recursos permite melhorar a utilização dos recursos compartilhados.

Algoritmo 2.1: Alocação de espaço em disco

```
   Entrada: dado, tamanhoDado, nó
1  se tamanhoDado ≤ tamanhoCache então
2      se espaçoDisponível < tamanhoDado então
3          se tamanhoDado > espaçoDisponível + tamanhoDadosExpirados então
4              espaçoDisponível ← espaçoDisponível + tamanhoDadosExpirados
5              se nóConsumidor = nóLocal então
6                  armazenaDadoLocal(dado, tamanhoDado);
7              senão
8                  armazenaDadoRemoto(dado, tamanhoDado, nó);
9              fim
10         senão
11             removeDadosExpirados(tamanhoDado)
12             armazena(dado)
13         fim
14     senão
15         armazena(dado)
16     fim
17 senão
18     rejeita(dado)
19 fim
```

Algoritmo 2.2: Função armazenaDadoLocal

```
   Entrada: dado, tamanhoDado
1  se tamanhoDado > espaçoDisponível + espaçoRemoto então
2      removeDadosRemotos()
3      espaçoDisponível ← espaçoDisponível + espaçoRemoto
4      enquanto tamanhoDado > espaçoDisponível e há mais dados locais a remover
       faça
5          remove dados do nó local
6          espaçoDisponível ← espaçoDisponível + tamanhoDadoRemovido
7      fim
8  senão
9      enquanto tamanhoDado > espaçoDisponível e há mais dados remotos a remover
       faça
10         remove dados do próximo nó com menor reputação
11         espaçoDisponível ← espaçoDisponível + tamanhoDadoRemovido
12     fim
13 fim
14 removeDadosExpirados()
15 armazena(dado)
```

Algoritmo 2.3: Função armazenaDadoRemoto

```
   Entrada: dado, tamanhoDado, nó_remoto
1  se tamanhoDado ≤ cota(nó_remoto) + espaçoDisponível então
2  |  enquanto tamanhoDado > espaçoDisponível + espaçoUtilizado(nó) e há mais
   |  dados de nós com cota excedida a remover faça
3  |  |  remove dados do próximo nó com menor reputação que esteja com cota
   |  |  excedida
4  |  |  espaçoDisponível ← espaçoDisponível + tamanhoDadoRemovido
5  |  fim
6  |  se tamanhoDado > espaçoDisponível então
7  |  |  remove dados de nó_remoto
8  |  fim
9  |  removeDadosExpirados()
10 |  armazena(dado)
11 senão
12 |  rejeita(dado)
13 fim
```

2.4 Considerações Finais

Neste capítulo foi apresentado o serviço de *cache* de disco para grades computacionais P2P que é composto pelo espaço em disco ocioso doado pelos nós que compõem a grade P2P. Por ser um serviço descentralizado e cada nó possuir autonomia para decidir quanto à alocação do espaço em disco, a *cache* prioriza a persistência dos dados submetidos por aplicações locais. O espaço não utilizado por essas aplicações é cedido para armazenamento dos dados provenientes de aplicações submetidas remotamente. A gestão desse espaço é gerida por um mecanismo baseado em reputação que divide o espaço em cotas proporcional à contribuição de cada nó. Com o intuito de melhorar a utilização da *cache*, os nós podem exceder a sua cota, desde que haja espaço ocioso em disco. Entretanto, a fração da *cache* consumida por nós excedentes poderá ser requisitada a qualquer momento por outros nós.

O mecanismo baseado em reputação proposto desencoraja o ato de *free-riding*, uma vez que a alocação do espaço em disco é definida pela reputação de um nó e a reputação de *free-riders* sempre será igual a zero. Deste modo, a única forma de se aumentar a cota de um nó, é aumentando sua reputação, e para aumentar sua reputação é necessário colaborar com o sistema. De fato, *free-riders* só poderão tirar proveito do sistema em momentos que não exista contenção de recursos.

Diferentemente de outras soluções encontradas na literatura, o mecanismo baseado em reputação descarta a necessidade de mecanismos de auditoria para avaliar a integridade de um dado, pois um favor só é contabilizado quando o dado é utilizado, e não quando armazenado.

Capítulo 3

Avaliação Experimental

A avaliação do serviço proposto neste trabalho consiste em demonstrar, através de experimentos de simulação, que *(i)* o serviço apresenta-se como uma solução viável para o suporte a aplicações BoT *data-intensive* que apresentam reuso de dados, *(ii)* o mecanismo baseado em reputação utiliza o espaço disponível para o sistema de *cache* em disco de forma justa e eficiente e *(iii)* quando o sistema entra em estado de contenção, *free-riders* são marginalizados em favor dos colaboradores.

O modelo de simulação adotado simplifica aspectos de uma grade P2P como, por exemplo, aqueles relativos à topologia e à falhas. Considera-se uma grade P2P composta por um número fixo de nós, com aplicação da rede de favores [6] para distribuição de recursos de processamento e do mecanismo de incentivo à colaboração descrito na Seção 2.3 para o gerenciamento do espaço de armazenamento compartilhado. O escalonamento das tarefas utiliza a heurística *Storage Affinity* [23], que associa as tarefas aos processadores de acordo com a afinidade entre eles. 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 tarefas 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 o impacto do uso do serviço de *cache* de disco para grades P2P, foi utilizada como métrica a taxa de acertos da *cache* em M*Byte* $\bar{H}$ (do inglês *cache MByte hit rate*), definida por:

$$\bar{H} = \frac{\sum_{i=1}^{T} MBytesEmCache(i)}{\sum_{i=1}^{T} MBytesRequeridos(i)} \tag{3.1}$$

onde T denota a quantidade de turnos, $MBytesRequeridos(i)$ denota a quantidade de M*bytes* requisitados por um conjunto de tarefas em um turno i e $MBytesEmCache(i)$ denota a quantidade de M*bytes* armazenada na *cache* de disco disponível para a execução das tarefas.

Para mostrar a eficiência e justiça do mecanismo, foi utilizado como métrica o nível de satisfação médio $\bar{A}$ em um turno T entre os nós, definido pela Equação 3.2. Esta métrica mede o nível de satisfação de um conjunto de nós que possuem perfis semelhantes, como por exemplo, um conjunto de nós *free-riders* ou colaboradores.

$$\bar{A}(T) = \frac{BP(T)}{BR(T)} \tag{3.2}$$

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 Modelagem da *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 os experimentos de simulação realizados neste trabalho são sintéticas e foram geradas a partir de um modelo matemático.

Uma *workload* W é constituída por um conjunto de tarefas BoT que são submetidas para execução em uma grade computacional. Neste modelo, uma tarefa é caracterizada pela quantidade de dados (em *bytes*) requerida pela mesma para sua execução. O tempo de execução

das tarefas não é considerado. As tarefas são distribuídas ao longo de turnos, onde a cada turno é associado um conjunto de dados. Formalmente, uma *workload* pode ser definida como:

$$W = \{d_j^i \mid 1 \leq i \leq T, 1 \leq j \leq N\} \tag{3.3}$$

onde T denota a quantidade de turnos, N denota a quantidade de nós que compõem a grade computacional P2P, e d_j^i representa o volume de dados que um nó $p_j \in P$ submete para execução na grade em um turno i. Desta forma, o volume de dados D submetido para execução em um turno i é definido por:

$$D_i = \sum_{j=1}^{N} d_j^i \tag{3.4}$$

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 3.1 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.

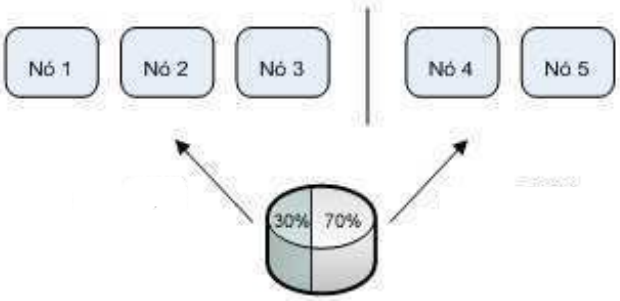


Figura 3.1: Balanceamento de carga entre os nós

Uma vez definidos os fatores de distribuição e balanceamento de carga entre os nós, cada d_j^i é obtido a partir da aplicação de uma distribuição uniforme dos dados baseada nos fatores de distribuição e balanceamento associados a uma heterogeneidade h_s para os nós. A heterogeneidade define a medida de variação para cada d_j^i, que pode assumir valores no intervalo $[0, 1]$. Formalmente, temos:

$$d_j^i = \begin{cases} U\left(\frac{\alpha \cdot D_i}{N \cdot \beta} \cdot \left(1 - \frac{h_s}{2}\right), \frac{\alpha \cdot D_i}{N \cdot \beta} \cdot \left(1 + \frac{h_s}{2}\right)\right), paraj \leq N \cdot \beta \\ \\ U\left(\frac{(1-\alpha) \cdot D_i}{N \cdot (1-\beta)} \cdot \left(1 - \frac{h_s}{2}\right), \frac{(1-\alpha) \cdot D_i}{N \cdot (1-\beta)} \cdot \left(1 + \frac{h_s}{2}\right)\right), paraj > N \cdot \beta \end{cases} \tag{3.5}$$

Uma tarefa especifica uma quantidade de *bytes* requeridos para execução. Nesse sentido, d_j^i representa uma quantidade de tarefas BoT *data-intensive* submetidas por um nó $p_j \in P$ para execução na grade P2P em um turno i. O conjunto de tarefas submetido por um nó a cada turno é composto por três subconjuntos de tarefas: *(i)* tarefas que reusam dados *inter-jobs*, *(ii)* tarefas que reusam dados *intra-job* e *(iii)* tarefas que utilizam novos dados.

O primeiro subconjunto de tarefas é composto por dados que já foram utilizados em turnos anteriores ou dados utilizados pelos demais nós no turno corrente, escolhidos de forma aleatória. A quantidade de *bytes* $d_{j_{inter}}^i$ de d_j^i reservada para esse tipo de tarefa é definida por:

$$d_{j_{inter}}^i = r_{inter} \cdot d_j^i \tag{3.6}$$

onde r_{inter} denota o grau de reuso *inter-jobs*, que pode assumir valores no intervalo $[0, 1]$.

Tarefas que reusam dados *intra-job* são aquelas que reusam dados gerados para o nó no turno corrente. Para esse subconjunto, a fração de *bytes* $d_{j_{intra}}^i$ é definida por:

$$d_{j_{intra}}^i = r_{intra} \cdot \left((1 - r_{inter}) \cdot d_j^i\right) \tag{3.7}$$

onde r_{intra} denota o grau de reuso *intra-job*, que pode assumir valores no intervalo $[0, 1]$.

Por fim, o terceiro subconjunto de tarefas é composto por novos dados gerados para o turno atual. A quantidade de *bytes* $d_{j_{novo}}^i$ alocada para esse conjunto é definida por:

$$d_{j_{novo}}^i = d_j^i - \left(d_{j_{inter}}^i + d_{j_{intra}}^i\right) \tag{3.8}$$

Dessa forma:

$$d_j^i = d_{j_{inter}}^i + d_{j_{intra}}^i + d_{j_{novo}}^i + \epsilon \tag{3.9}$$

Nesta equação, a soma das três quantidades apresentadas pode ser inferior ao valor de d_j^i devido ao erro de aproximação de valores (ϵ). Como exemplo, seja $d_{j_{inter}}^i = 100$ M*Bytes*

a quantidade de dados que deve ser reusada *inter-jobs*. Suponha que em um determinado instante t, já tenham sido selecionados $99.9MBytes$ de dados e que não existe dado inferior a $200KBytes$. Desta forma, no instante $t + 1$, nenhum dado satisfaz a lacuna restante. Para respeitar o limite imposto por $d^i_{j_{inter}}$ o procedimento é encerrado neste ponto com um erro $\epsilon = 0.1$ M*Bytes*.

A geração das tarefas t^k_j é regida pela seguinte variável aleatória uniforme:

$$t^k_j = U\left(g_t \cdot \left(1 - \frac{h_t}{2}\right), g_t \cdot \left(1 + \frac{h_t}{2}\right)\right) \tag{3.10}$$

onde g_t representa a granularidade da tarefa e h_t o fator de heterogeneidade para a mesma. A granularidade refere-se ao tamanho dos dados da aplicação, enquanto que a heterogeneidade das tarefas corresponde à variação do tamanho desses dados.

3.2 Avaliação do Impacto do Uso de *Cache* em Disco

A avaliação do impacto do uso de *cache* em disco visa apresentar evidências de que o serviço proposto neste trabalho viabiliza o suporte a aplicações BoT *data-intensive* que apresentam reuso de dados, reduzindo a quantidade de transferências desses dados entre domínios administrativos.

3.2.1 Descrição dos Cenários

Os experimentos de simulação realizados consideram o uso de quatro estratégias de *cache* para substituição de dados. A aplicação dessas estratégias possibilitou avaliar a eficiência do sistema de *cache* de disco para grades P2P ao reduzir a quantidade de movimentação de dados entre os nós. A seguir são apresentadas as descrições das quatro estratégias utilizadas segundo Podlipnig e Böszörmenyl [21]:

- *LRU (Least Recently Used)*: baseia-se no princípio da localidade temporal dos dados, isto é, se um dado é referenciado, ele tende a ser referenciado novamente em um curto intervalo de tempo. Esta estratégia classifica os dados pelo último valor de referência, iniciando a substituição pelos dados que possuem o menor valor.

- *LFU (Least Frequently Used)*: baseia-se na popularidade dos dados, ou seja, um dado que possui popularidade elevada também possui altos índices de freqüência. Esta estratégia classifica os dados de acordo com a freqüência de uso dos mesmos. O processo de substituição considera, em princípio, os dados que apresentam menores valores de freqüência. Para simplificar o modelo de simulação, foi adotado, neste trabalho, a implementação *In-Cache LFU* [21], onde o valor da freqüência dos dados não é mantido quando o mesmo é removido da *cache*.

- *LFU-DA (LFU with Dynamic Aging)*: baseia-se no mesmo princípio da estratégia LFU, acrescida de um fator de envelhecimento que se propõe a melhorar a eficiência das *caches*. Esse fator previne que dados que possuem elevados valores de freqüência, mas não são mais usados, sejam mantidos em *cache*. Para tanto, a cada nova requisição de alocação de espaço em disco, a estratégia (re)calcula o valor V de cada dado armazenado na *cache* da seguinte forma:

$$V = f + L \tag{3.11}$$

  onde f denota o valor da freqüência e L o fator de envelhecimento (inicializado com zero). O processo de substituição dos dados considera a escolha do dado com menor V. Quando um dado é removido da *cache*, o valor V deste dado é atribuído a L. Desta forma, novos dados inseridos no sistema terão o fator de envelhecimento agregado ao valor da freqüência.

- *GDSF (Greedy-Dual Size with Frequency)*: esta estratégia considera, além da freqüência e o fator de envelhecimento, o tamanho dos dados. De maneira análoga à estratégia LFU-DA, a cada nova requisição de alocação de espaço em disco, a estratégia (re)calcula o valor V de acordo com a seguinte equação:

$$V = f \cdot \frac{c}{s} + L \tag{3.12}$$

  onde s denota o tamanho do dado em *bytes* e c é uma constante. Nos experimentos de simulação, foi considerado $c = 1$ com o intuito de maximizar o *hit rate*.

Os experimentos foram realizados sob cinco configurações de grade P2P que diferem no tamanho total do espaço em disco disponível para o sistema de *cache* de disco, sendo estes 64 G*bytes*, 128 G*bytes*, 256 G*bytes*, 512 G*bytes* e 1.024 G*bytes*. Em todos os experimentos foi considerada uma quantidade fixa de 16 nós. Foi gerado um conjunto de *workloads* que contempla 81 tipos de *workloads* distintas, definidos de acordo com os valores especificados na Tabela 3.1.

Variável	**Valores**
Quantidade de nós (N)	16
Fator de distribuição de carga (α)	10%, 30% e 50%
Fator de balanceamento (β)	25%, 50% e 75%
Reuso *intra-job* (r_{intra})	25%, 50% e 75%
Reuso *inter-jobs* (r_{inter})	25%, 50% e 75%
Heterogeneidade do nó (h_s)	25%
Heterogeneidade da tarefa (h_t)	25%
Granularidade da tarefa (g_t)	100 K*bytes*, 10 M*bytes* e 1 G*byte*
Tamanho total da *cache*	64 G*bytes*, 128 G*bytes*, 256 G*bytes*, 512 G*bytes* e 1.024 G*bytes*

Tabela 3.1: Características do cenário para avaliação do impacto do uso de *cache* em disco

Foram realizados aproximadamente 2.400 experimentos de simulação para cada estratégia, que foram suficientes para a obtenção de resultados com erro inferior a 5% para mais ou para menos, com nível de confiança de 95%.

3.2.2 Resultados e Análise

Nesta subseção são apresentados os resultados obtidos para a avaliação do serviço de *cache* em disco para grades P2P realizado através de experimentos de simulação para os cenários descritos na subseção anterior. Os gráficos apresentam em seu eixo vertical a taxa de acertos da *cache* em M*Byte* (vide Equação 3.1) para as quatro estratégias utilizadas nos experimentos. O eixo horizontal apresenta os diferentes tamanhos do espaço total para o sistema de *cache* de disco. Os dados utilizados para a geração dos gráficos estão disponíveis no Apêndice A.

A Figura 3.2 apresenta o resultado dos experimentos de simulação para valores distintos da taxa de reuso *intra-jobs* (r_{intra}). Para $r_{intra} = 25\%$ (Figura 3.2(a)), a *cache* apresenta uma redução em torno de 20% da quantidade de M*Bytes* a serem transferidos para execução das aplicações. Devido à baixa taxa de reuso de dados *intra-job*, o ganho obtido com o aumento

do tamanho da *cache* é mínimo, uma vez que a cada turno uma grande massa de novos dados é inserida no sistema. Por outro lado, para $r_{intra} = 50\%$ e 75%, Figuras 3.2(b) e 3.2(c), respectivamente, o sistema reduz em até 60% a quantidade de M*Bytes* a serem transferidos entre os nós. Esse comportamento é justificado pela alta taxa de reuso de dados em um mesmo conjunto de tarefas. Nesses três cenários, observa-se que, à medida que o espaço total para armazenamento em *cache* aumenta, maior é o índice de acertos.

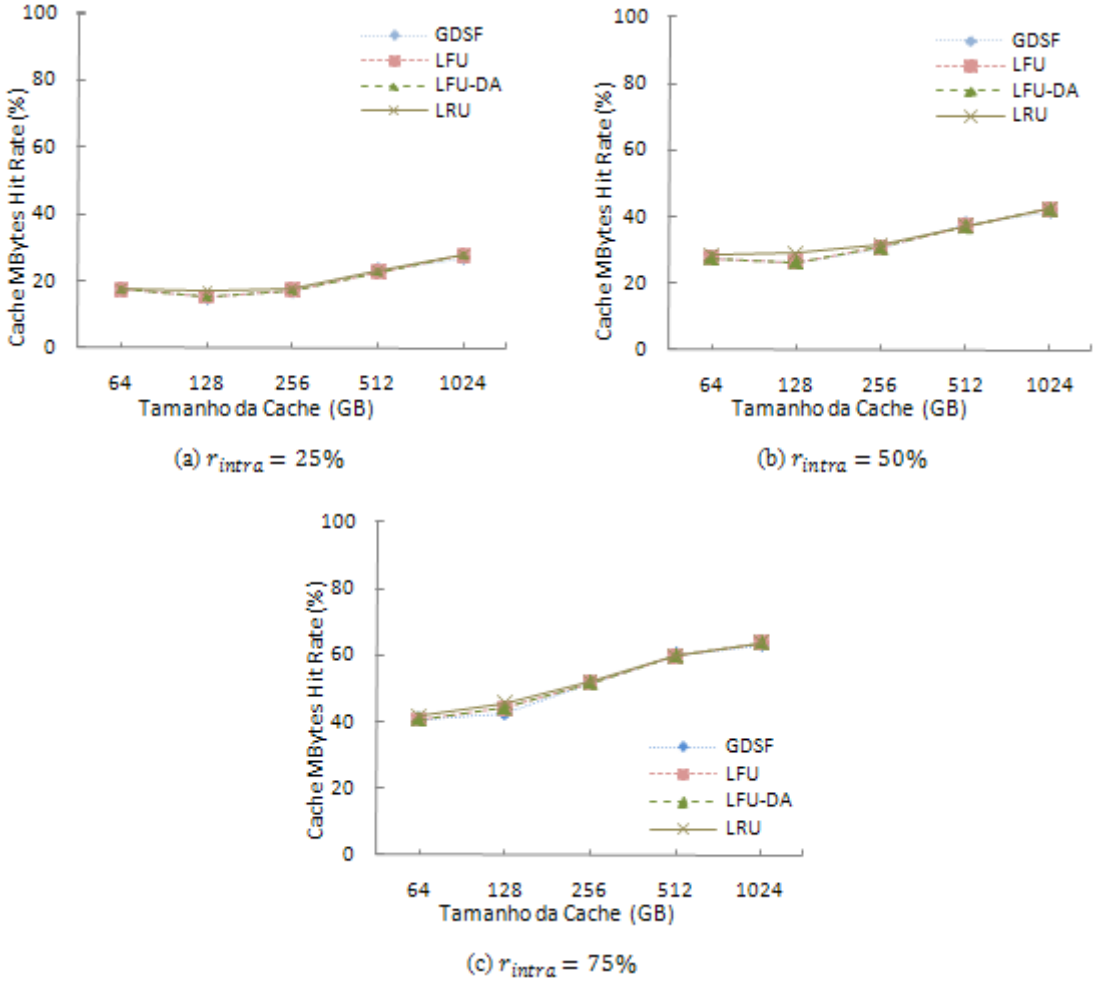


Figura 3.2: Taxa de acertos da *cache* em M*Bytes* para diferentes valores de reuso *intra-job* (r_{intra})

De maneira análoga a análise anterior, a Figura 3.3 mostra o resultado dos experimentos de simulação para distintos valores da taxa de reuso *inter-jobs* (r_{inter}). Para $r_{inter} = 25\%$ (Figura 3.3(a)), a taxa de acertos se mantém em torno de 35%. Como a taxa de reuso é baixa, praticamente não existe ganho quando o tamanho da *cache* é aumentado, diferentemente do que ocorre na análise anterior, onde o ganho mínimo é conseqüência do reuso dos dados

em um mesmo turno. Para $r_{inter} = 50\%$ e 75%, Figuras 3.3(b) e 3.3(c), respectivamente, a taxa de acertos cresce à medida que o espaço em disco disponível para o sistema de *cache* aumenta. Da mesma forma que no reuso *intra-job*, quanto maior for a taxa de reuso, maior será a eficiência do sistema de *cache* em disco.

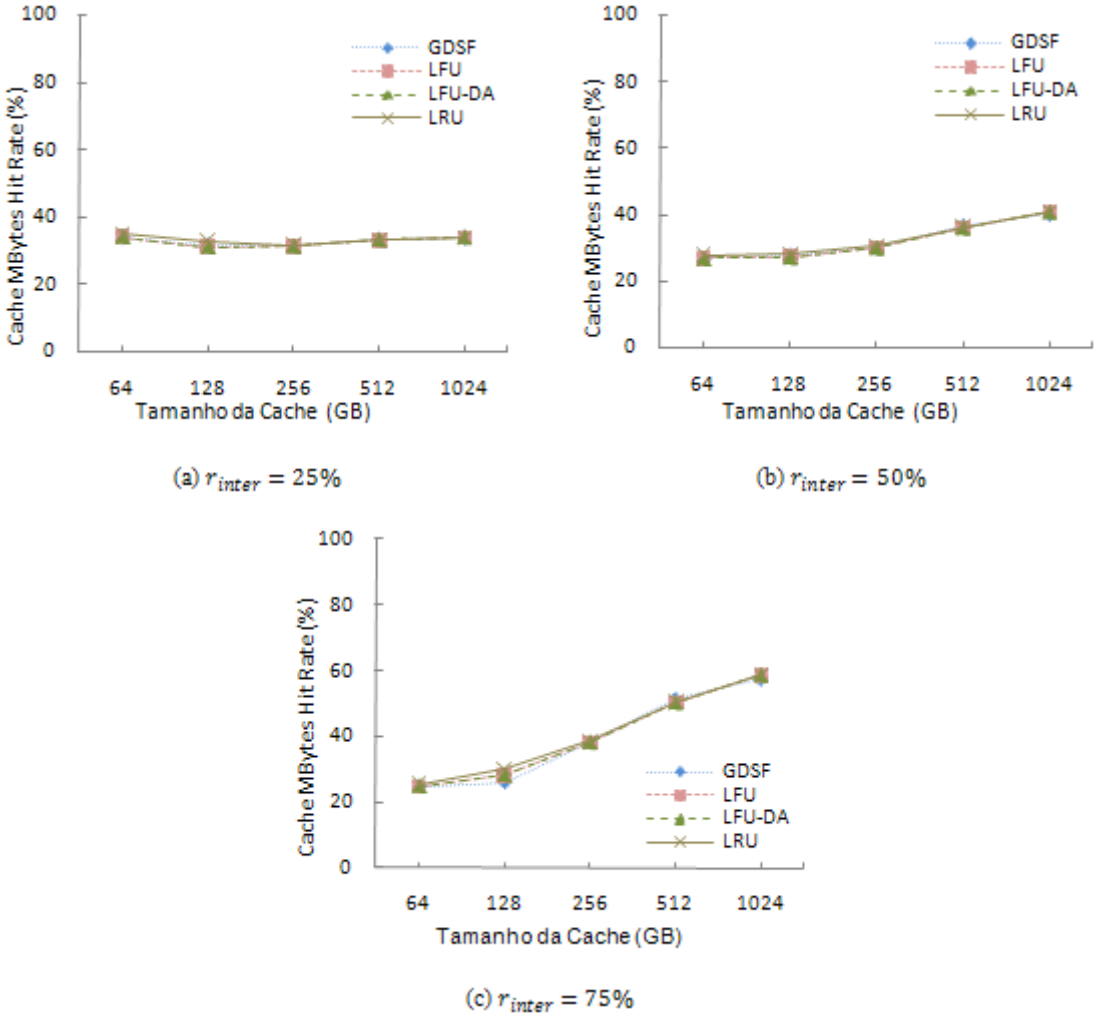


Figura 3.3: Taxa de acertos da *cache* em M*Bytes* para diferentes valores de reuso *inter-jobs* (r_{inter})

Os resultados apresentados demonstram que o conjunto de *workloads* adotado para a realização dos experimentos de simulação gerado a partir do modelo matemático descrito na Seção 3.1, é suficientemente genérico e contempla uma parcela representativa de *workloads* para aplicações BoT *data-intensive*, de forma que podemos observar que as estratégias de substituição adotadas apresentam o mesmo comportamento no sistema. Para demonstrar as vantagens de uma determinada estratégia, seria necessário escolher *workloads* características para cada estratégia.

A Figura 3.4 apresenta a taxa de acertos da *cache* para duas configurações específicas: *(i)* $r_{intra} = 75\%$ e $r_{inter} = 25\%$ (Figura 3.4(a)), e *(ii)* $r_{intra} = 25\%$ e $r_{inter} = 75\%$ (Figura 3.4(b)). Ambas as configurações apresentam uma distribuição homogênea dos dados ($\alpha = 50\%$ e $\beta = 50\%$), com fatores de heterogeneidade iguais a 25%. A primeira configuração favorece a estratégia *LRU*, uma vez que essa *workload* possui o princípio da localidade de dados como característica, i.e., alta taxa de reuso de dados *intra-job*. Por outro lado, a segunda configuração favorece a estratégia *LFU* devido ao alto grau de reuso de dados *inter-jobs*.

As simulações realizadas demonstram que o serviço de *cache* de disco para grades computacionais P2P proposto neste trabalho reduz a quantidade de transferências de dados entre domínios administrativos, independentemente da estratégia de substituição de dados adotada para uso no sistema de *cache* em disco. Através dos experimentos de simulação mostrou-se, também que, quanto maior for a taxa de reuso de dados *intra-job* e/ou *inter-jobs*, maior será a utilidade do sistema de *cache* em disco.

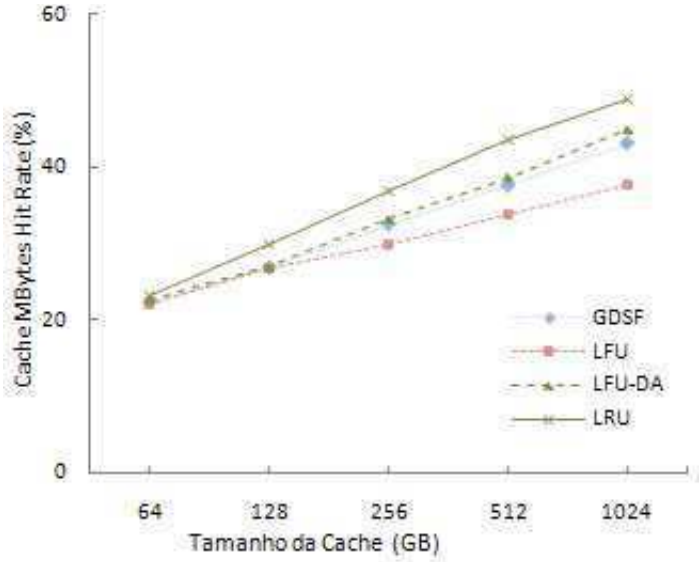
3.3 Eficiência e Justiça

A avaliação do mecanismo de reputação proposto neste trabalho consiste em mostrar que o mecanismo provê alocação justa e eficiente do espaço disponível para o sistema de *cache* e que *free-riders* são marginalizados quando o sistema entra em estado de contenção, i.e., não existe espaço em disco suficiente para persistência dos dados.

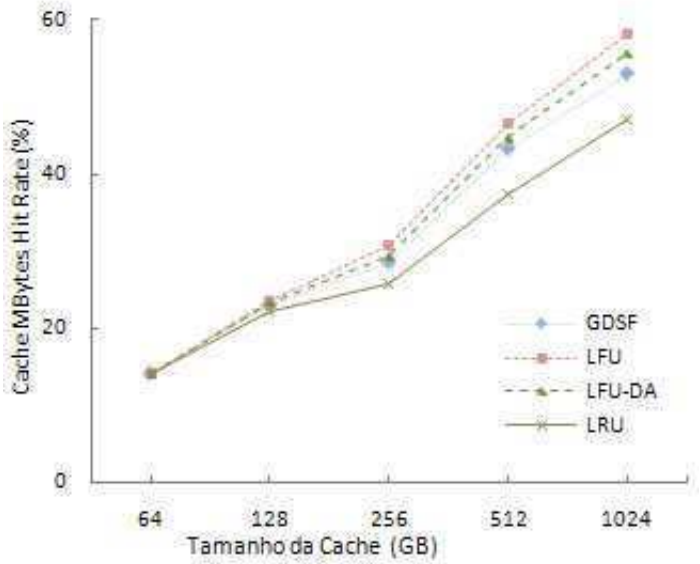
3.3.1 Descrição dos Cenários

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 mostrar 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* de 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



(a) $r_{intra} = 75\%$ e $r_{inter} = 25\%$



(b) $r_{intra} = 25\%$ e $r_{inter} = 75\%$

Figura 3.4: Taxa de acertos da *cache* em M*Bytes* para duas configurações específicas

workloads pelo fator de balanceamento de carga (β), sendo a carga distribuída simetricamente entre os fragmentos ($\alpha = 50\%$). 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 fator de balanceamento de carga 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 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. A Tabela 3.2 apresenta um sumário das variáveis que caracterizam o primeiro e o segundo grupo.

Variável	**Valores (Grupo 1)**	**Valores (Grupo 2)**
Quantidade de nós (N)	100	
Quantidade de *free-riders* (f)	50%	25%, 50% e 75%
Fator de distribuição de carga (α)	50%	
Fator de balanceamento (β)	25%, 50% e 75%	50%
Reuso *intra-job* (r_{intra})	25%, 50% e 75%	
Reuso *inter-jobs* (r_{inter})	25%, 50% e 75%	
Heterogeneidade do nó (h_s)	25%	
Heterogeneidade da tarefa (h_t)	25%	
Granularidade da tarefa (g_t)	100 K*bytes*, 10 M*bytes* e 1 G*byte*	
Tamanho total da *cache*	300 G*bytes* e $2,1$ T*bytes*	

Tabela 3.2: Características do primeiro cenário para avaliação da alocação justa e eficiente do sistema de *cache*

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* de 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 ($\alpha = 50\%$). Neste cenário, todos os nós possuem o papel de colaboradores. Foram consideradas situações

de alta e média contenção, onde média contenção implica em um espaço em disco disponível para *caching* de $1,3$ T*bytes*, representando cerca de 40% do volume total de dados, e alta contenção implica em um espaço em disco de 300 G*bytes*, que representa em torno de 10% do volume total de dados. A Tabela 3.3 apresenta um sumário das variáveis que caracterizam este cenário.

Variável	**Valores (Grupo 1)**	**Valores (Grupo 2)**	**Valores (Grupo 3)**
Quantidade de nós (N)	100		
Fator de distribuição de carga (α)	50%		
Fator de balanceamento (β)	25%	50%	75%
Reuso *intra-job* (r_{intra})	25%, 50% e 75%		
Reuso *inter-jobs* (r_{inter})	25%, 50% e 75%		
Heterogeneidade do nó (h_s)	25%		
Heterogeneidade da tarefa (h_t)	25%		
Granularidade da tarefa (g_t)	100 K*bytes*, 10 M*bytes* e 1 G*byte*		
Tamanho total da *cache*	300 G*bytes* e $1,3$ T*bytes*		

Tabela 3.3: Características do segundo cenário para avaliação da alocação justa e eficiente do sistema de *cache*

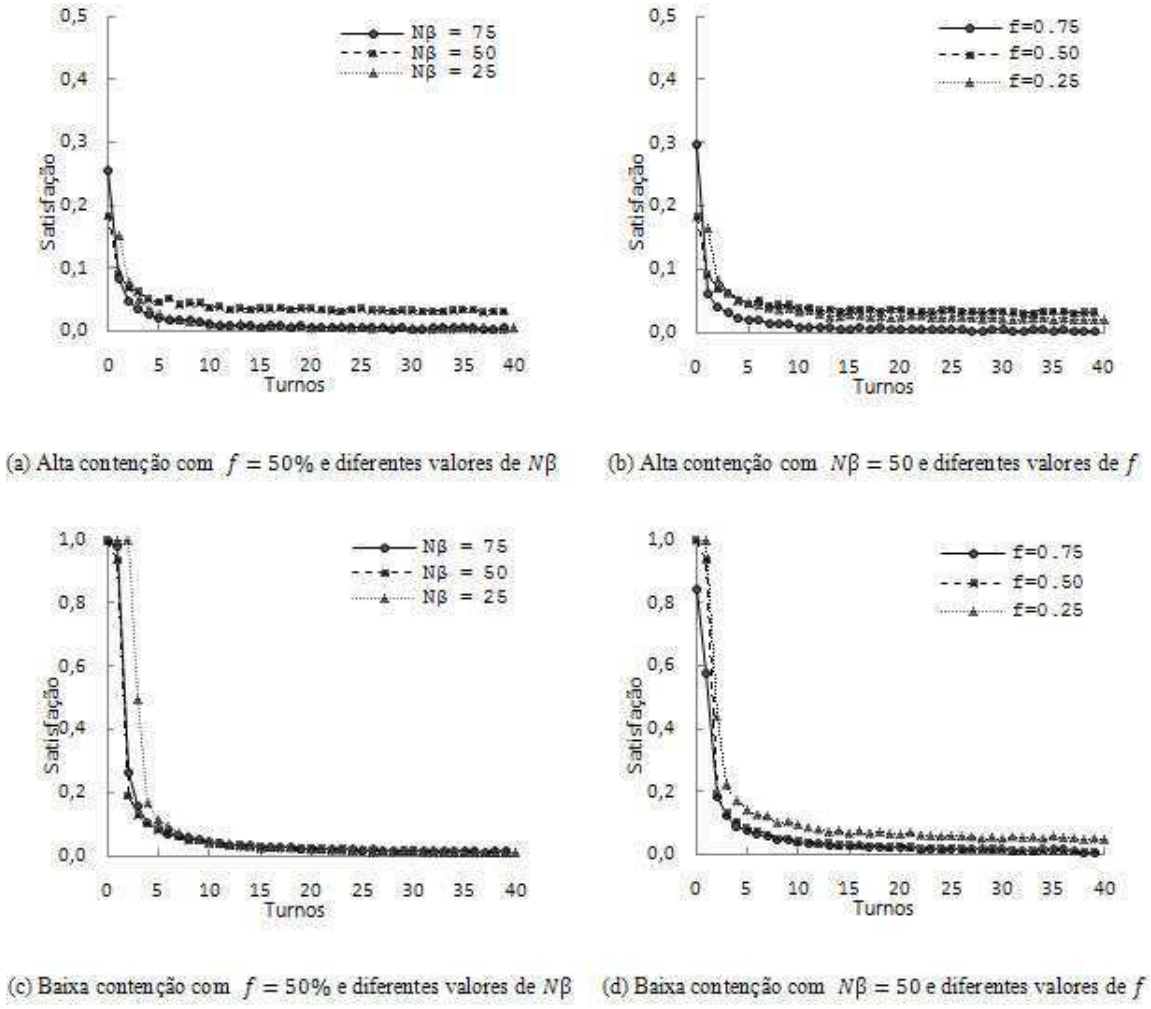
Para o primeiro cenário foram realizados em torno de 300 experimentos de simulação, e para o segundo foram realizados aproximadamente 4.000 experimentos de simulação, que foram suficientes para a obtenção de resultados com erro inferior a 5% para mais ou para menos, com nível de confiança de 95%.

3.3.2 Resultados e Análise

Nesta subseção são apresentados 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 3.2) entre os nós no decorrer dos turnos (eixo horizontal).

A Figura 3.5 apresenta os resultados para o primeiro cenário, sendo os dois primeiros gráficos (Figuras 3.5(a) e 3.5(b)) relacionados à execução em estado de alta contenção e os seguintes (Figuras 3.5(c) e 3.5(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* de 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.



(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 3.5: Nível de satisfação médio para *free-riders*

Em momento de alta contenção (Figuras 3.5(a) e 3.5(b)) a marginalização dos *free-riders* ocorre quase que imediatamente, devido à alta demanda dos colaboradores por espaço em disco. Na Figura 3.5(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 16% da carga total. Como todo o restante da carga é submetida pelos

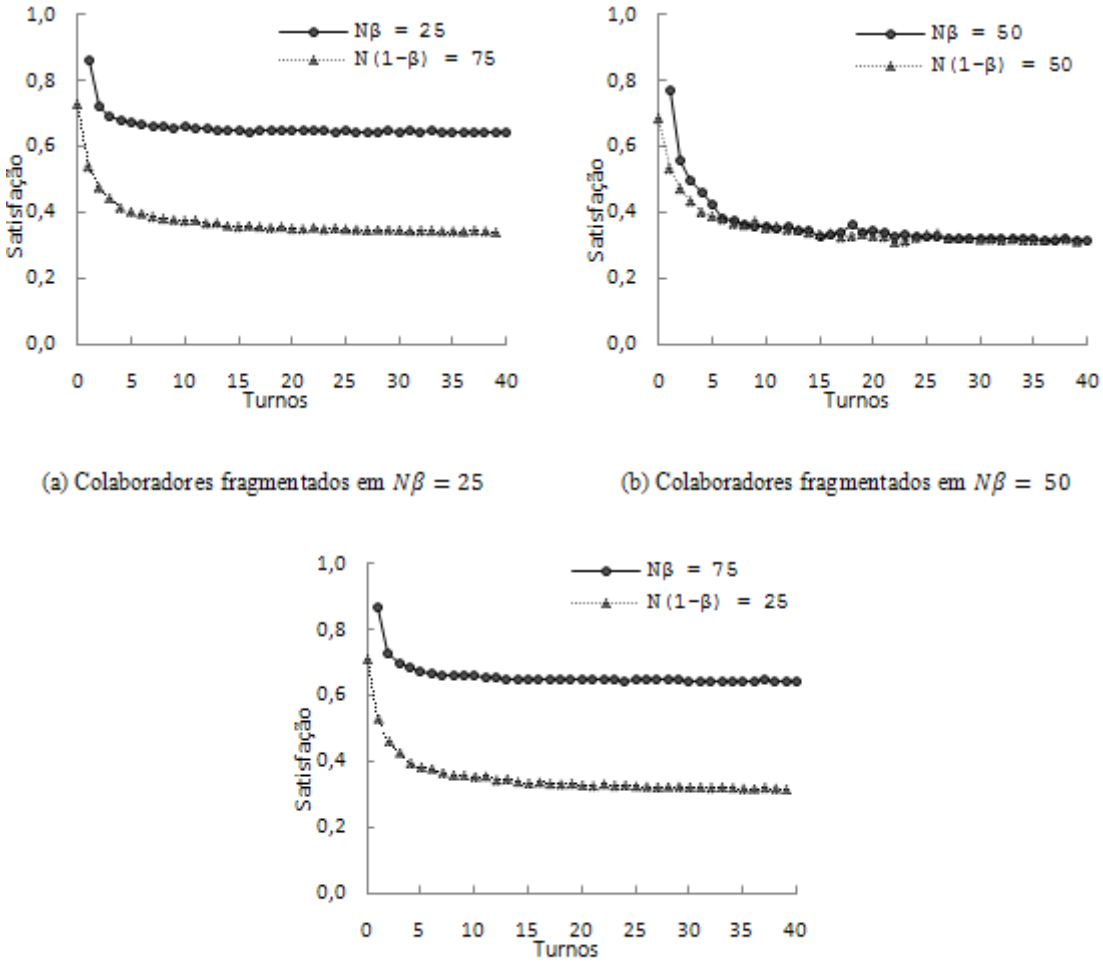
colaboradores, os espaços disponíveis nas *caches* são alocados para armazenamento de dados provenientes destes. A falta de espaço ocioso conduz o mecanismo à marginalização de nós que não contribuem com a grade. Para $N \cdot \beta = 50$, o nível de satisfação apresenta o menor índice de marginalização, 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 3.5(b), mostra 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 (Figuras 3.5(c) e 3.5(d)), mostra 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 Figura 3.6 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 3.6(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 3.6(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 3.6(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

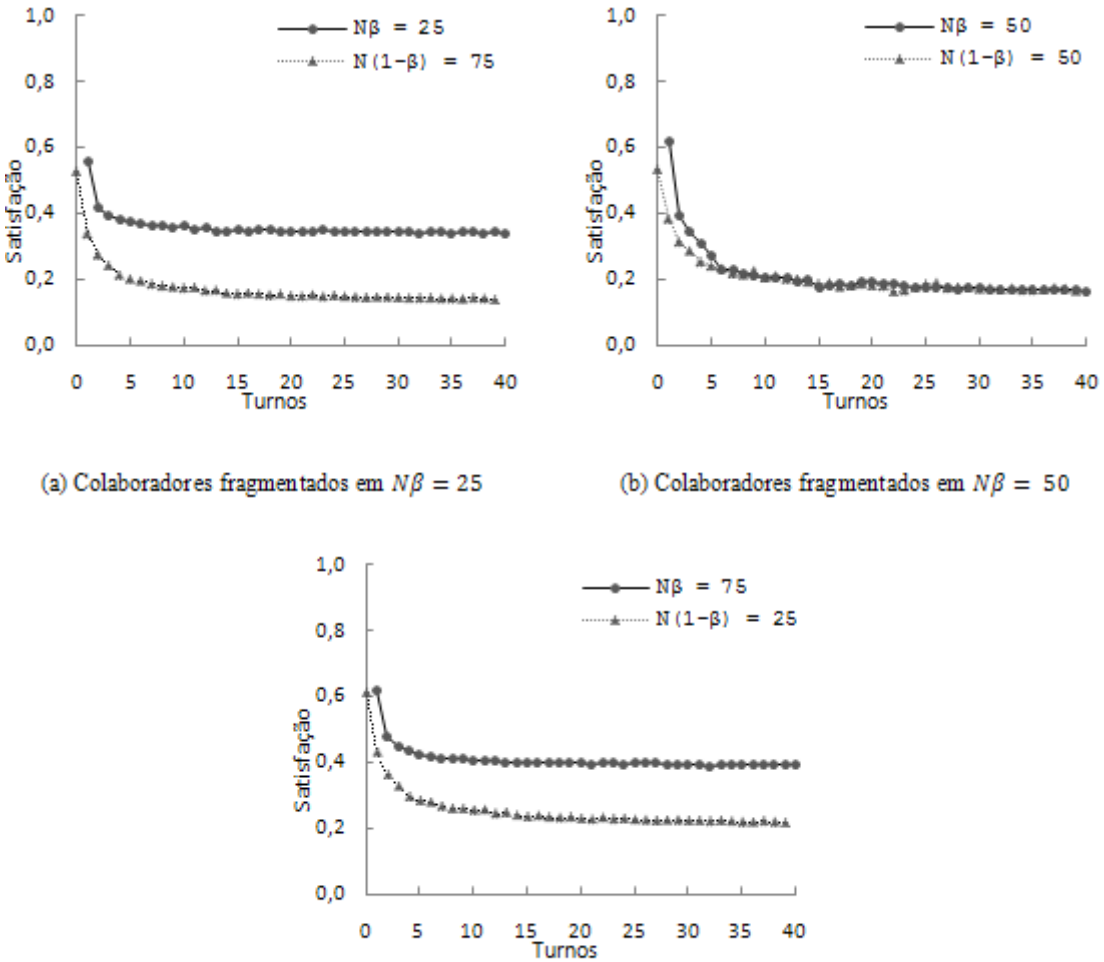


(a) Colaboradores fragmentados em $N\beta = 25$

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

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

Figura 3.6: Nível de satisfação médio para colaboradores (média contenção)



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

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

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

Figura 3.7: Nível de satisfação médio para colaboradores (alta contenção)

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 3.6(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 zero, o que já era esperado haja vista que o nosso modelo divide o espaço ocioso igualitariamente entre nós que possuem reputação nula.

De maneira análoga à análise anterior, a Figura 3.7 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 apresentado na Figura 3.6, diferindo apenas no grau de satisfação dos colaboradores.

3.4 Escalabilidade

O estudo da escalabilidade visa demonstrar que o mecanismo baseado em reputação mostra-se eficiente quando em ambientes com grandes quantidades de nós. Neste trabalho, foi considerada a quantidade fixa de 1.000 e 10.000 nós para os experimentos.

3.4.1 Descrição dos Cenários

A fim de avaliar a escalabilidade do mecanismo, foram considerados três tipos de cenários que diferem pela quantidade de nós *free-riders* envolvidos na grade. As proporções de *free-riders* para cada cenário constituem 25%, 50% e 75% da totalidade de nós. As *workloads* utilizadas foram geradas com carga distribuída simetricamente entre os nós ($\alpha = 50\%$, $\beta = 50\%$, sendo $N \cdot \beta = 500$ ou 5.000), com taxas de reuso de dados *intra-job* e *inter-jobs* iguais a 25%.

Cada nó colaborador contribui com 28 G*bytes* de espaço em disco para a grade. Desta forma, no primeiro cenário, as *caches* possuem um total de 21 T*bytes* para cenários com 1.000 nós, ou 210 T*bytes* para cenários compostos por 10.000 nós, no segundo 14 T*bytes* ou 140 T*bytes* e no terceiro 7 T*bytes* ou 70 T*bytes*, que representam aproximadamente 70%,

45% e 20% do volume total de dados, respectivamente. A Tabela 3.4 apresenta um resumo das variáveis que caracterizam os cenários deste experimento.

Variável	**Valores (Cenário 1)**	**Valores (Cenário 2)**	**Valores (Cenário 3)**
Quantidade de nós (N)	1.000 e 10.000		
Quantidade de *free-riders* (f)	25%	50%	75%
Fator de distribuição de carga (α)	50%		
Fator de balanceamento (β)	50%		
Reuso *intra-job* (r_{intra})	25%		
Reuso *inter-jobs* (r_{inter})	25%		
Heterogeneidade do nó (h_s)	25%		
Heterogeneidade da tarefa (h_t)	25%		
Granularidade da tarefa (g_t)	100 K*bytes*, 10 M*bytes* e 1 G*byte*		
Tamanho total da *cache* (1.000 nós)	7 T*bytes*, 14 T*bytes* e 21 T*bytes*		
Tamanho total da *cache* (10.000 nós)	70 T*bytes*, 140 T*bytes* e 210 T*bytes*		

Tabela 3.4: Características dos cenários para avaliação da escalabilidade

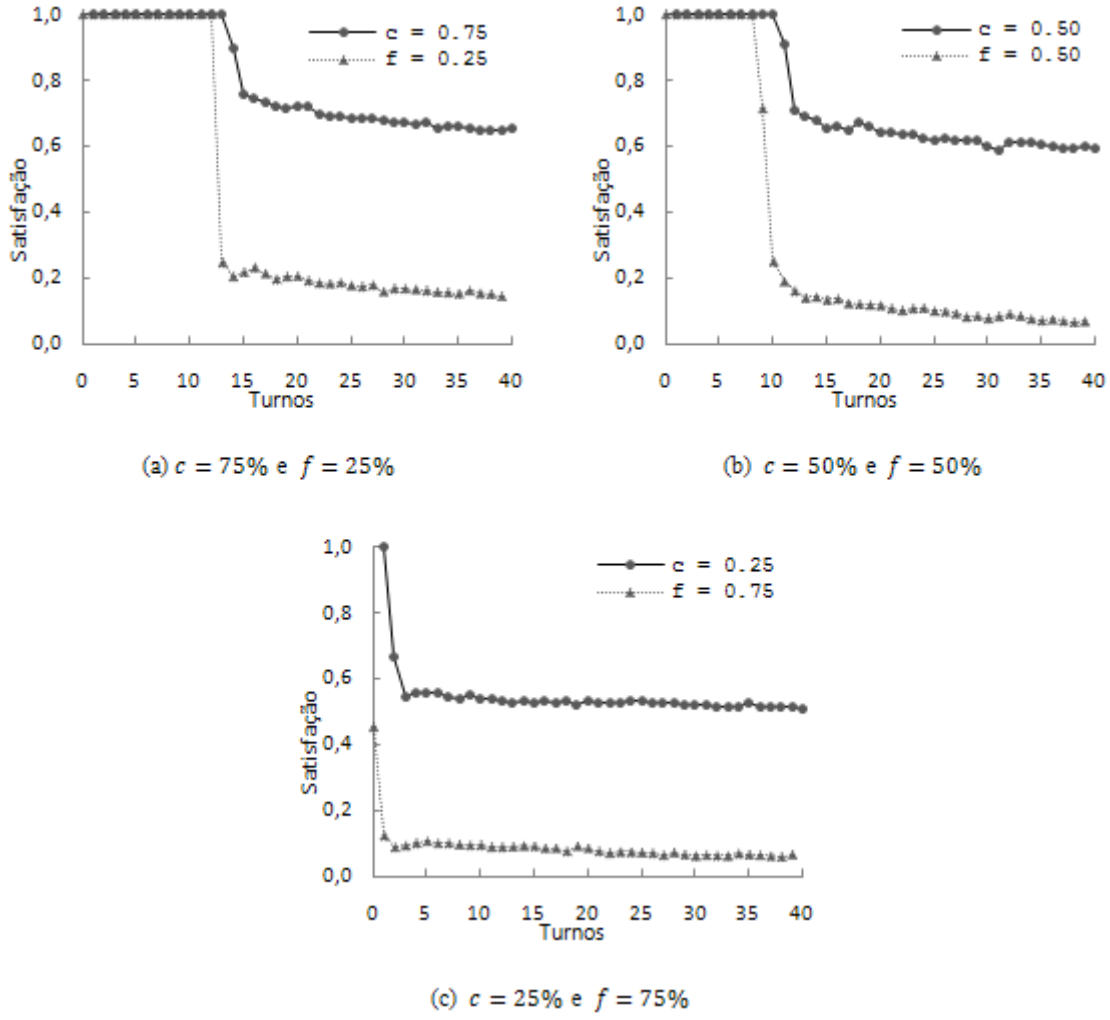
No primeiro cenário objetiva-se analisar o comportamento do mecanismo em momentos de baixa contenção. Já no segundo, objetiva-se analisar o mecanismo em momentos de média contenção de recursos. Por fim, no terceiro cenário, pretende-se observar como o mecanismo atua em momentos de alta contenção, onde a quantidade de nós colaboradores é significativamente menor do que a de *free-riders*.

Para os experimentos com 1.000 nós foram realizadas aproximadamente 90 simulações, e para os experimentos com 10.000 nós foram realizados 75 experimentos de simulação, que foram suficientes para obter-se resultados com nível de confiança de 95% e erro inferior a 5% para mais ou para menos.

3.4.2 Resultados e Análise

A Figura 3.8 apresenta os resultados para os experimentos de simulação com 1.000 nós. Na Figura 3.8(a), onde 25% dos nós são *free-riders*, a demanda por espaço em disco nos primeiros 13 turnos é completamente satisfeita tanto para os colaboradores (c) como para os *free-riders* (f). Entretanto, à medida que o sistema começa a entrar em estado de contenção, o mecanismo privilegia os nós colaboradores, marginalizando rapidamente os *free-riders*. É importante ressaltar que o nível de satisfação obtido pelos *free-riders* neste cenário é devido ao consumo do espaço ocioso da *cache* pelas tarefas que requerem dados de pequena ou média granularidade (100 K*bytes* e 1 M*byte* respectivamente).

Para $f = 50\%$ (Figura 3.8(b)), o mecanismo apresenta resultados similares ao cenário anterior. Em momentos em que não há contenção, ambos os grupos de nós são satisfeitos. Entretanto, a partir do turno 9, o sistema entra em estado de contenção, ou seja, não existe espaço suficiente em disco para armazenamento dos dados. Novamente, nós *free-riders* são marginalizados em favor dos nós colaboradores. Como a quantidade de nós colaboradores é reduzida neste cenário, o nível de satisfação dos *free-riders* também é menor, uma vez que existe uma quantidade reduzida de recursos ociosos.



(a) $c = 75\%$ e $f = 25\%$

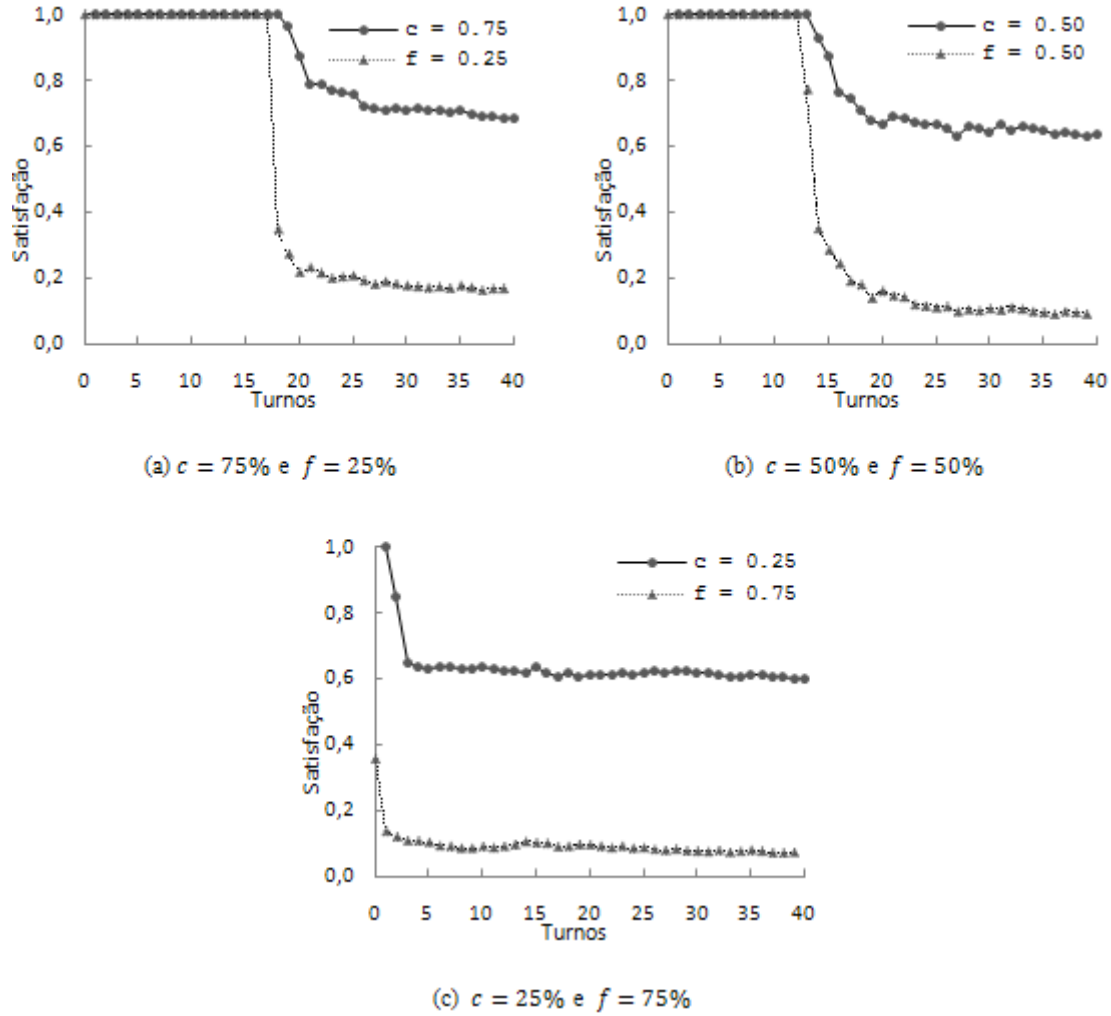
(b) $c = 50\%$ e $f = 50\%$

(c) $c = 25\%$ e $f = 75\%$

Figura 3.8: Nível de satisfação médio para colaboradores (c) e *free-riders* (f) para 1.000 nós

Em momentos de alta contenção (Figura 3.8(c)), o nível de satisfação dos nós atinge o menor valor, o que já era esperado. Como a quantidade de nós colaboradores é a menor de todos os cenários e o conjunto de *free-riders* o maior, o espaço ocioso em *cache* é praticamente nulo, o que leva o nível de satisfação dos *free-riders* para menos de 8%. Esse nível de

satisfação é alcançado, em quase sua totalidade, apenas por tarefas que requerem dados de pequena granularidade.



(a) $c = 75\%$ e $f = 25\%$

(b) $c = 50\%$ e $f = 50\%$

(c) $c = 25\%$ e $f = 75\%$

Figura 3.9: Nível de satisfação médio para colaboradores (c) e *free-riders* (f) para 10.000 nós

A Figura 3.9 apresenta os resultados dos experimentos para os cenários com 10.000 nós. De maneira análoga aos resultados encontrados para 1.000 nós, o mecanismo apresenta-se eficaz quanto à marginalização dos *free-riders* em momentos de contenção de recursos. Novamente, é importante ressaltar que o mecanismo apenas atua em momentos de contenção. Desta forma, quando há espaço ocioso em disco, nós *free-riders* se beneficiam deste espaço para processamento de suas aplicações, como pode ser observado nos primeiros turnos das Figuras 3.9(a) e 3.9(b). Por outro lado, em momentos de alta contenção (Figura 3.9(c)), praticamente não existe espaço ocioso, neste caso, os *free-riders* são marginalizados logo nos primeiros turnos.

3.5 Considerações Finais

As simulações realizadas demonstraram que o serviço de *cache* de disco para grades computacionais P2P proposto neste trabalho, viabiliza o suporte à aplicações BoT *data-intensive* ao reduzir a quantidade de transferências de dados entre domínios administrativos, e que o mecanismo do sistema provê alocação justa e eficiente do espaço em disco compartilhado pelos nós, marginalizando *free-riders* em momentos de contenção de recursos. É importante ressaltar que, em todos os cenários simulados na presença de *free-riders* e em momentos de contenção de recursos, a marginalização destes ocorre quase que imediatamente independente da quantidade de nós envolvidos na grade. Esse comportamento é justificado pela atuação do mecanismo de reputação baseado em cotas que, em momentos de contenção, define a fração de espaço em disco que cada nó terá para armazenamento no sistema de *cache*, baseado no histórico de favores ofertados e recebidos. Em momentos em que não há contenção de recursos, o mecanismo não atua, e o espaço ocioso em disco pode ser consumido pelos *free-riders*. Desta forma, a eficiência da *cache* é aumentada.

Os experimentos também demonstraram que, independente da quantidade de nós que compõem a grade, o mecanismo mostra-se eficiente quanto à alocação de recursos e marginalização de *free-riders* em prol de nós colaboradores. Por ser um serviço descentralizado e cada nó ser autônomo em relação à definição das cotas, a escalabilidade do serviço não agrega problemas de *overhead* de comunicação.

O simulador desenvolvido, assim como o gerador sintético de *workloads* baseado no modelo matemático descrito na Seção 3.1, estão disponíveis para *download* no seguinte endereço: *http://redmine/projects/list_files/ogstorageelement*.

Capítulo 4

Estudo de Caso

Neste capítulo é apresentado um estudo de caso realizado em uma grade computacional P2P em produção. Através desse estudo, objetiva-se avaliar a eficiência do sistema de *cache* de disco quando submetido ao processamento de uma aplicação BoT *data-intensive*. Através da realização de experimentos, espera-se demonstrar que o espaço em disco doado para armazenamento seja gerenciado de forma eficiente, de modo a reduzir a quantidade de transferências de dados entre domínios administrativos. Por conseqüência, espera-se que o tempo de processamento das tarefas que compõem uma aplicação também seja reduzido.

4.1 OurGrid

O *OurGrid* [8] é um *middleware* aberto e de fácil implantação para grades computacionais entre-pares onde os participantes compartilham recursos ociosos de processamento e armazenamento de dados, para a execução de aplicações que possuam tarefas independentes, i.e., aplicações *Bag-of-Tasks*.

A Figura 4.1 apresenta a arquitetura do *OurGrid* com seus quatro componentes principais: o *broker*, que é uma interface cliente de submissão de tarefas; o *peer*, entidade que agrupa consumidores e recursos de um domínio administrativo, controlando a doação e alocação desses recursos; o *worker*, recurso executor das tarefas submetidas pelo usuário; e o *discovery service*, um serviço de descoberta de recursos disponíveis na grade [11]. A comunicação entre os quatro componentes é realizada através do *Commune* [18], um *middleware* que utiliza o protocolo XMPP [3] (*eXtensible Messaging and Presence Protocol*) para troca

de informações. A composição de cada domínio administrativo é denominada *site* (ou sítio), onde cada *site* possui um único *peer* com um ou mais *brokers* e vários *workers*.

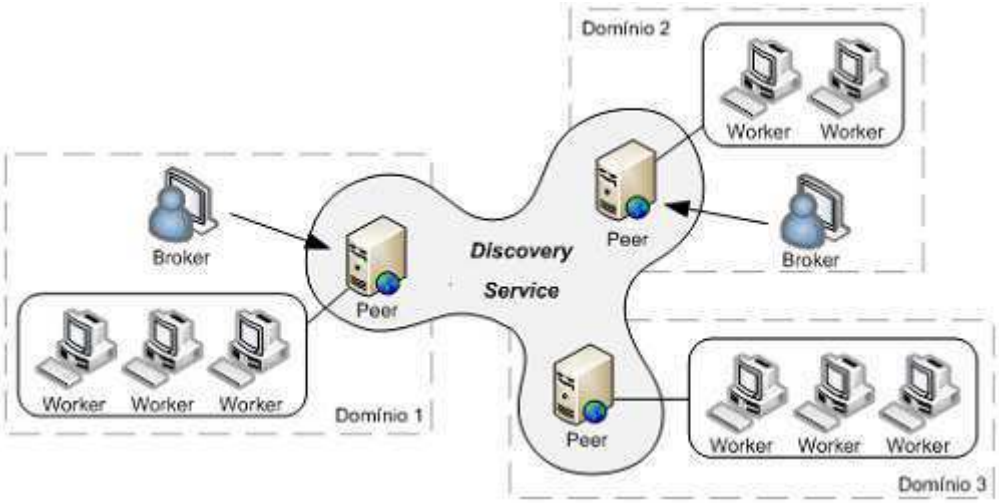


Figura 4.1: Arquitetura do *OurGrid*

O *OurGrid* utiliza a rede de favores proposta por Andrade et al. [6]. A rede de favores atua como um mecanismo de incentivo à colaboração de recursos computacionais de processamento, desencorajando o ato de *free-riding*. A alocação desses recursos é baseada na reputação dos nós. Nesse sentido, a reputação de um nó y em um nó z é obtida a partir da contabilidade dos favores ofertados por z a y (poder de processamento $\times$ tempo de doação) e dos favores recebidos por z de y (poder de processamento $\times$ tempo de consumo).

A persistência de dados no *OurGrid* é realizada através de dois mecanismos de armazenamento: *(i)* o *storage* e *(ii)* o *playpen*. O primeiro armazena um dado de forma persistente (mas não garantida, i.e., *best-effort*) em uma estação de trabalho (*worker*), enquanto que o segundo é um mecanismo de armazenamento temporário que persiste os dados apenas durante a execução da tarefa. Apesar do *OurGrid* persistir dados de forma a serem reutilizados em futuras execuções, ele não apresenta mecanismos de gerência do espaço em disco doado para o armazenamento destes. Em momentos de contenção desses recursos, os dados não poderão mais ser armazenados na área destinada para o *storage*. A cada nova execução de uma tarefa os novos dados deverão ser transferidos para o *playpen*, com exceção dos que já se encontram armazenados no *storage*.

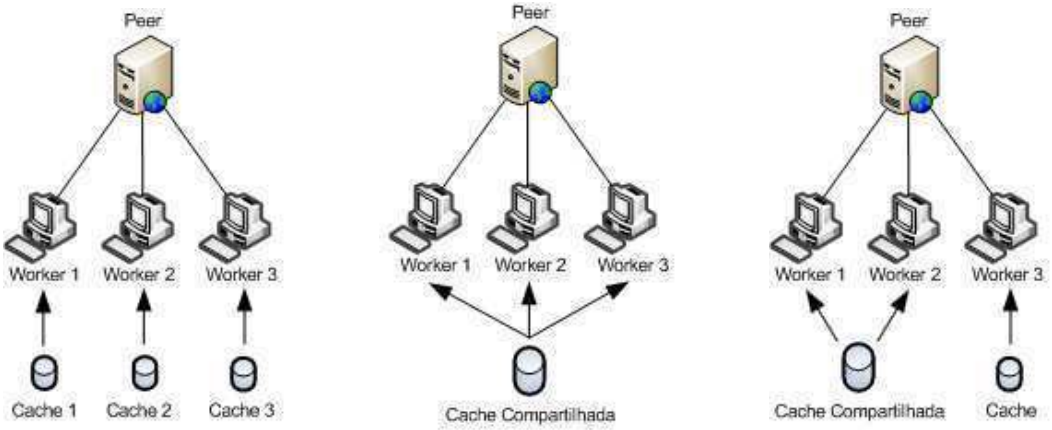
Outra deficiência apresentada pelo *OurGrid* é que o sistema é desprovido de um mecanismo que incentive os nós envolvidos na grade a doarem seus espaços em disco ociosos

para armazenamento. Desta forma, o suporte à aplicações BoT *data-intensive* no *OurGrid* é também limitada por aspectos de disponibilidade de espaço para armazenamento.

4.2 Implementação do Protótipo do Serviço de *Cache*

A concepção do protótipo do serviço de *cache* de disco para o *middleware OurGrid* consistiu na implementação do mecanismo de incentivo à colaboração e adaptação do *playpen* para uma área de *caching*.

O serviço de *cache* de disco pode ser implantado através de três abordagens: utilizando-se uma *(i) cache* individual (Figura 4.2(a)), uma *(ii) cache* compartilhada (Figura 4.2(b)) ou uma *cache* híbrida (Figura 4.2(c)). Na primeira, cada *worker* é responsável por gerenciar o seu próprio espaço em disco disponibilizado para armazenamento. Na *cache* compartilhada, a área de armazenamento é provida por um sistema de compartilhamento de dados, como por exemplo, o NFS (*Network File System*) [2]. Desta forma, todos os *workers* pertencentes a um mesmo domínio têm acesso ao mesmo sistema de arquivos. A terceira forma de configurar um *site* é através de uma abordagem híbrida, onde um domínio é composto por uma ou mais *caches* individuais e uma ou mais *caches* compartilhadas.



(a) *Cache* individual (b) *Cache* compartilhada (c) *Cache* híbrida

Figura 4.2: Formas de realizar *caching* no *OurGrid*

A adoção de um sistema de arquivos compartilhado pode acarretar problemas de concorrência. Nesse sentido, foi desenvolvido um componente para gerenciamento da *cache* compartilhada, denominado *shared service*, que tem por objetivo tratar as requisições oriundas dos *workers* para persistência de dados na *cache* compartilhada. Considerando-se que

os dados utilizados em uma grade P2P são *write-once*, i.e., só são escritos uma única vez, o serviço apenas endereça problemas de concorrência na primeira escrita do dado.

Quando uma nova requisição de armazenamento chega ao *shared service*, ele verifica se existe alguma outra requisição já em processamento para o mesmo dado. Caso exista uma requisição em processamento proveniente de outro *worker*, o serviço encaminha a nova requisição a uma fila de espera. Caso contrário, o serviço inicia o procedimento de verificação de disponibilidade de armazenamento descrito no Algoritmo 2.1 (vide Seção 2.3). Assim que o processo de transferência de um dado para a *cache* compartilhada é completado, uma mensagem é enviada para todos os *workers* que realizaram uma requisição para o mesmo dado, informando que o dado se encontra disponível em *cache*, e portanto, não necessita ser transferido. Se ocorrerem problemas ou cancelamento da transferência do dado, o *shared service* verifica se existe alguma outra requisição para o mesmo dado na fila de espera. Caso exista, um novo procedimento de transferência é iniciado. A Figura 4.3 apresenta um diagrama de atividades que descreve esse procedimento.

O gerenciamento dos favores ofertados e recebidos por um *site* é realizado pelo *peer*. Quando um *worker* ou o *shared service* necessita calcular a cota de um nó, uma solicitação da contabilidade dos favores é realizada ao *peer* local. O *peer* contabiliza os favores e informa a reputação do nó (vide Equação 2.4). Provido do valor da reputação, o *worker* pode calcular a cota estimada para o nó remoto (vide Equação 2.5).

No momento em que um *worker* termina a execução de uma tarefa, ele envia uma requisição de contabilização de favor ofertado ao *peer* local, que o contabiliza como negativo. Quando o *broker* responsável pela submissão da tarefa constata que a mesma foi finalizada com sucesso, ele envia uma requisição de favor recebido ao seu *peer* local, que o contabiliza como positivo. Esse procedimento só é realizado entre domínios administrativos. Quando o *broker* e o *worker* pertencem ao mesmo domínio, não existe a contabilização de favores, uma vez que o serviço de *cache* prioriza o armazenamento dos dados locais.

4.3 Experimentação

A avaliação experimental consiste em demonstrar que o tempo de processamento de uma aplicação BoT *data-intensive* em grades P2P é reduzido em conseqüência da redução da

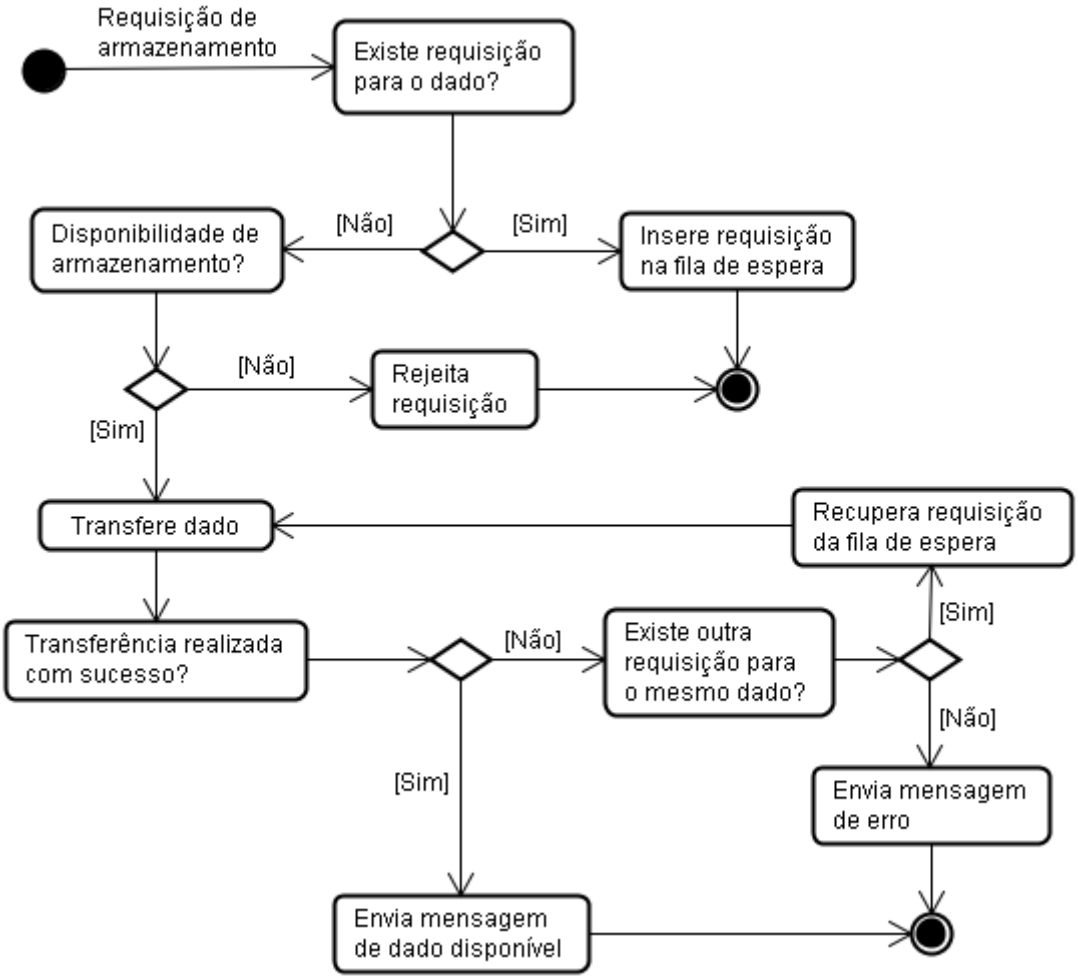


Figura 4.3: Gerenciamento de requisições para a *cache* compartilhada

quantidade de transferências realizadas entre domínios administrativos. Através dos experimentos objetiva-se, também, mostrar que o sistema de *cache* utiliza o espaço em disco doado para armazenamento de forma justa e eficiente, marginalizando *free-riders* em momentos de contenção de recursos.

Apesar do *OurGrid* prover suporte a duas formas de implantação do sistema de *cache*, os experimentos realizados neste trabalho consideram o uso da *cache* compartilhada, pois acredita-se que a mesma apresenta maior impacto com relação à quantidade de transferências.

Para demonstrar a eficiência do sistema de *cache* de disco, foram utilizadas duas métricas: *(i)* o *makespan* e *(ii)* o nível de satisfação (vide Equação 3.2) entre nós colaboradores e *free-riders*. O *makespan* mede a diferença de tempo entre o início e o fim da execução de um conjunto de tarefas que compõem uma aplicação. A partir da medição do *makespan*, objetiva-se avaliar a economia de tempo de processamento de tarefas BoT *data-intensive* em cenários que implementam o serviço de *cache* proposto neste trabalho.

4.3.1 Descrição da Aplicação

O *Meta-Dock* [26] é uma aplicação designada para estudos farmacêuticos desenvolvida pelo Instituto de Biotecnologia da Universidade Nacional Autônoma do México (UNAM). A aplicação consiste em identificar um ligante (uma molécula) que seja capaz de inibir a ação de uma proteína responsável pela proliferação de uma enfermidade. Através de um programa de *docking*, especificamente o *AutoDock* [1], são realizadas simulações que predizem a orientação mais provável de um ligante com relação a uma proteína para formar um complexo estável. A execução da aplicação segue o seguinte *workflow*: de posse de uma proteína e um ligante, o usuário especifica mapeamentos (possíveis orientações) entre essas moléculas, e as envia para execução na grade. Ao término da execução, o usuário pode verificar qual a orientação mais provável que forma um complexo estável.

4.3.2 Descrição dos Cenários

Os experimentos realizados consistem na submissão de um conjunto de tarefas BoT *data-intensive*, onde cada tarefa requer como dados de entrada o programa a ser executado, uma

proteína e um ligante com seu respectivo mapeamento. A base de moléculas utilizada nos experimentos foi obtida através do portal ZINC [4]. Foram considerados dois tipos de cenários, sendo o primeiro caracterizado pela medição do *makespan* para as duas estratégias de armazenamento providas pelo *OurGrid* (*playpen* e *storage*) e a estratégia de *caching*, e o segundo caracterizado pela presença de *free-riders* na grade.

O primeiro cenário é composto por dois domínios administrativos, sendo o primeiro composto de quatro recursos de processamento e uma *cache* compartilhada utilizando-se NFS, e o segundo composto por um *broker* que realiza a submissão das tarefas para execução na grade. Neste cenário foi utilizada uma aplicação que visa identificar, dentre quatro ligantes, qual o que produz um complexo mais estável para uma determinada proteína. A Tabela 4.1 apresenta o tamanho dos dados de entrada utilizados no experimento. Neste cenário, todas as execuções foram realizadas sem contenção de recursos.

Dado de Entrada	**Tamanho (M*bytes*)**
Programa (B)	$0,554$
Proteína 01 (Pr_1)	$9,352$
Ligante 01 (Li_1)	$9,127$
Ligante 02 (Li_2)	$8,963$
Ligante 03 (Li_3)	$8,861$
Ligante 04 (Li_4)	$9,032$

Tabela 4.1: Dados de entrada para o primeiro cenário

O segundo cenário é composto por três *sites* dispostos da seguinte forma: o primeiro dispõe de quatro recursos de processamento e uma *cache* compartilhada utilizando-se NFS; o segundo e terceiro são compostos por um *broker*, cada, que realizam submissão de tarefas para execução na grade. Entretanto, para o *site* provedor de recursos, o segundo *site* é visto como colaborador, enquanto que o terceiro como *free-rider*. Neste cenário, cada *broker* submete uma aplicação que visa identificar dentre dois ligantes, qual o mais apropriado para uma proteína. Diferentemente do cenário anterior, o tamanho da *cache* é limitado a 60 M*bytes*. A Tabela 4.2 apresenta os dados utilizados para a execução deste cenário.

Dado de Entrada	**Tamanho (M*bytes*)**
Programa (B)	$0,554$
Proteína 01 (Pr_1)	$9,352$
Proteína 02 (Pr_2)	$9,092$
Ligante 01 (Li_1)	$9,127$
Ligante 02 (Li_2)	$8,963$
Ligante 03 (Li_3)	$8,861$
Ligante 04 (Li_4)	$9,032$

(a) Colaborador

Dado de Entrada	**Tamanho (M*bytes*)**
Programa (B)	$0,554$
Proteína 03 (Pr_3)	$10,007$
Proteína 04 (Pr_4)	$8,567$
Ligante 05 (Li_5)	$8,689$
Ligante 06 (Li_6)	$8,734$
Ligante 07 (Li_7)	$8,451$
Ligante 08 (Li_8)	$7,629$

(b) *Free-rider*

Tabela 4.2: Dados de entrada para o segundo cenário

Para o primeiro cenário foram realizadas 180 execuções, e para o segundo foram realizadas 45 execuções, que foram suficientes para a obtenção de resultados com nível de confiança de 95% e erro inferior a 5% para mais ou para menos.

4.3.3 Resultados e Análise

A Figura 4.4 apresenta o resultado dos experimentos realizados para medir o impacto do uso do serviço de *cache* no *OurGrid* em comparação ao uso das estratégias de armazenamento *playpen* e *storage*. O gráfico apresenta em seu eixo vertical a medição do *makespan* da aplicação para cada estratégia ao longo do tempo (eixo horizontal). Como esperado, a adoção do *playpen* é a estratégia menos eficiente, uma vez que os dados sempre serão transferidos para a área de armazenamento. Por outro lado, a estratégia de *storage*, como também a de *cache*, apresentam redução considerável no tempo de execução da aplicação (em torno de 40%), sendo praticamente iguais a partir da segunda execução, pois os dados estão disponíveis na área de armazenamento, descartando a necessidade de transferência. Entretanto, a estratégia *storage* apresenta-se menos eficiente no momento da primeira execução da aplicação devido à necessidade da transferência dos dados para todos os *workers*, fracionando a banda para realização de operações de transferências de um mesmo dado. Com a utilização de uma *cache* compartilhada, os dados que apresentam reuso são transferidos uma única vez para um *worker*. Enquanto o dado é transferido, os demais *workers* que apresentaram solicitação para o mesmo dado ficam no aguardo do fim da operação. Deste modo, a banda é dedicada a realização desta operação, que, por conseqüência, implica na redução do tempo de transferência. Outra limitação apresentada pela estratégia *storage* é que a mesma é desprovida de um mecanismo de gerência automatizada do espaço, ou seja, uma vez que o espaço disponível para

armazenamento for consumido, os novos dados enviados para execução na grade serão transferidos para o *playpen*. Desta forma, em momentos de contenção de recursos a utilidade da estratégia *storage* é limitada apenas ao reuso dos dados previamente armazenados.

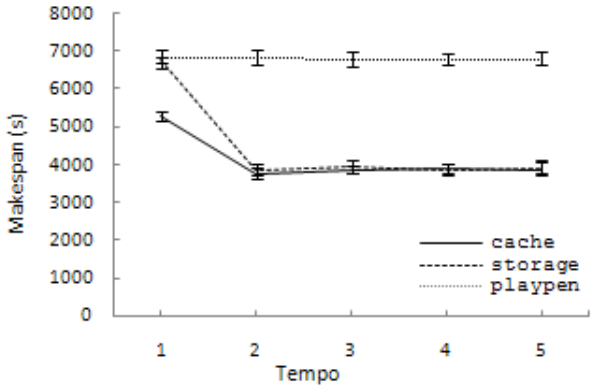
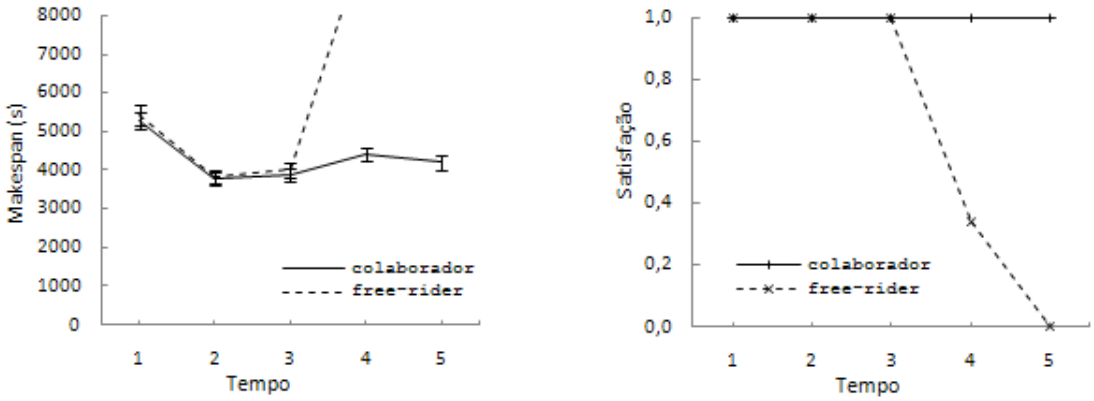


Figura 4.4: Comparação entre estratégias de armazenamento

A Figura 4.5 mostra os resultados dos experimentos para avaliação do mecanismo de reputação e marginalização de *free-riders* no *OurGrid*. Seja $k = \{B, Pr, Li\}$ uma tarefa, onde B denota o programa a ser executado, Pr uma proteína e Li um ligante. No instante de tempo t_1, o nó colaborador submete um conjunto de tarefas $T_c(t_1) = \{\{B, Pr_1, Li_1\}, \{B, Pr_1, Li_2\}\}$ e o nó *free-rider* um conjunto $T_f(t_1) = \{\{B, Pr_3, Li_5\}, \{B, Pr_3, Li_6\}\}$, de modo que praticamente todo o espaço da *cache* é ocupado. Nos instantes t_2 e t_3 o mesmo conjunto de tarefas é submetido, de modo que ambos os nós são satisfeitos. Entretanto, no instante t_4, o nó colaborador submete um conjunto $T_c(t_4) = \{\{B, Pr_2, Li_2\}, \{B, Pr_2, Li_3\}\}$ para execução na grade. Como a *cache* entra em estado de contenção, dados pertencentes ao nó *free-rider* são removidos para prover espaço para o nó colaborador. Desta forma, quando o nó *free-rider* tenta executar sua aplicação, apenas uma das três moléculas requeridas para execução encontra-se disponível. No instante de tempo t_5, o nó colaborador submete um novo conjunto de tarefas $T_c(t_5) = \{\{B, Pr_2, Li_3\}, \{B, Pr_2, Li_4\}\}$ para execução. Novamente, a única molécula remanescente do nó *free-rider* é removida para o armazenamento do ligante Li_4. A partir deste instante, independente de qualquer conjunto de tarefas submetido pelo nó *free-rider*, seu nível de satisfação será sempre igual a zero. Para que o nó possa voltar a usufruir da grade, ele deve começar a colaborar com o sistema.



(a) (b)

Figura 4.5: Medição do *makespan* (a) e do nível de satisfação (b) entre colaborador e *free-rider*

4.4 Considerações Finais

Neste capítulo foi apresentado um estudo de caso em uma grade computacional P2P em produção para validação do serviço de *cache* de disco. O serviço foi implementado em uma versão do *middleware OurGrid* e experimentado com uma aplicação real.

Os experimentos demonstraram que o uso do serviço de *cache* de disco reduziu o tempo de execução das aplicações avaliadas nos cenários avaliados em aproximadamente 20% quando comparado à primeira execução com a área de *storage*, e em torno de 40% quando comparado a execução utilizando o *playpen*. Os experimentos também demonstraram que em momentos em que não há contenção de recursos, todos os nós usufruem do serviço, enquanto que em momentos de contenção, *free-riders* são marginalizados em favor de nós colaboradores.

Capítulo 5

Trabalhos Relacionados

Neste capítulo é apresentada a contextualização deste trabalho em relação ao estado da arte em sistemas de *cache* de disco para grades P2P, como também mecanismos de incentivo à colaboração, sendo ressaltadas as principais deficiências e diferenças destes em relação ao serviço apresentado.

5.1 Sistemas P2P de Armazenamento *versus Caching* em Grades P2P

Sistemas P2P de armazenamento têm por objetivo prover um sistema global de armazenamento de dados, que forneça um serviço de alta confiabilidade e disponibilidade dos dados. Esse tipo de sistema é caracterizado pelo uso de mecanismos de replicação, tolerância a falhas e segurança. Por outro lado, sistemas de *cache* de disco para grades P2P se propõem a maximizar o desempenho das aplicações, através da redução da quantidade de transferências de dados entre domínios administrativos. Diferentemente dos sistemas P2P de armazenamento, em sistemas de *cache* não existe garantias de que o dado persistido estará disponível quando requisitado. Apesar dos sistemas divergirem quanto a seus propósitos, ambos apresentam desafios em comum como, por exemplo, *(i)* a alocação justa e eficiente do espaço em disco disponível para armazenamento e *(ii)* o incentivo à colaboração.

Por ser desconhecida a existência de trabalhos que se utilizam de um sistema de *cache* de disco para armazenamento de dados em grades P2P, e como os sistemas P2P de armazena-

mento endereçam desafios em comum ao tipo de sistema proposto, este trabalho inspira-se em soluções propostas para sistemas P2P de armazenamento.

5.2 Mecanismos de Incentivo à Colaboração

Uma das grandes contribuições 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. [15] classificam os métodos de incentivo à colaboração e combate a *free-riding* baseados em três abordagens: monetária, reciprocidade e reputação. 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 das interações 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.

Golle et al. [13], foram um dos primeiros autores a propor a abordagem monetária em sistemas P2P de armazenamento. Desde então, diversos esquemas monetários foram propostos no contexto de redes de compartilhamento de dados [10]. Entretanto, esse tipo de abordagem possui um custo elevado para manutenção, uma vez que ela requer uma infraestrutura dedicada para realização do serviço de contabilidade e pagamentos.

Pollack et al. [22], propuseram um mecanismo de cotas baseado em *vouchers* emitidos por uma entidade central. Um *voucher* define a quantidade de espaço em *bytes* que um nó poderá armazenar no sistema por um determinado intervalo de tempo. Entretanto, o mecanismo não permite que *vouchers* sejam renovados, ou seja, quando um *voucher* expira e necessita-se do dado armazenado, uma nova solicitação de armazenamento deve ser reali-

zada. O uso de entidades centralizadas agrega outros problemas ao sistema, tais como a inserção de um único ponto de falha e *overhead* de comunicação na rede.

Ngan et al. [19] 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. Ma e Wang [17] apresentam em seu trabalho um mecanismo descentralizado de incentivo à colaboração baseado em contratos entre os nós. Nesse mecanismo, 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. Tanto a solução apresentada por Ngan et al. como a apresentada por Ma e Wang podem subutilizar os recursos de armazenamento, uma vez que espaços que não estão sendo utilizados por um nó, não poderão ser aproveitados por outros nós da comunidade. Esse tipo de solução pode ser encontrada em sistemas como o *Samsara* [9] e o *PeerStore* [16].

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 adoção de uma solução descentralizada para mecanismos de incentivo parece ser mais viável. Entretanto, esse tipo de arquitetura agrega alguns novos desafios, tais como ataques *white-washing* provenientes de *free-riders*. Nesse tipo de ataque, *free-riders* adquirem constantemente novas identidades a fim de se livrarem de baixas reputações. Uma possível solução para conter esses ataques, proposta por Castro et al. [7], é a adoção de identificações únicas emitidas por uma entidade certificadora. Entretanto, o custo para manter essa infra-estrutura é elevado.

O mecanismo de *caching* proposto 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*) [6]. 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 um 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. A fim de prevenir ataques

white-washing, a reputação de um nó nunca se torna negativa. Desta forma, *free-riders* são desencorajados a ficarem trocando suas identidades. 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* [25] 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 o mecanismo proposto neste trabalho não necessita 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.

5.3 Considerações Finais

Neste capítulo foram apresentados alguns mecanimos de incentivo à colaboração e alocação justa e eficiente do espaço em disco para sistemas P2P. Dentre as soluções apresentadas, este trabalho inspira-se na rede de favores como mecanismo descentralizado de incentivo à colaboração, e no mecanismo de cotas elásticas proposto pelo *Solomon* para maximização da utilização do espaço.

Capítulo 6

Conclusões 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 o desempenho desse tipo de aplicação, quando executadas 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* de 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 foi apresentado um mecanismo de gerência de *cache* de 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 de 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 melhora da utilização dos recursos. Em estado de alta contenção, a alocação do espaço disponível se baseia em um mecanismo de reputação inspirado na rede de favores proposta por Andrade et al. [6].

A avaliação do mecanismo, realizada através de experimentos de simulação, mostra que

o mecanismo de *cache* de disco proposto neste trabalho viabiliza o suporte à aplicações BoT *data-intensive*, provendo eficiência, justiça e escalabilidade para o sistema de *cache* de disco para grades computacionais entre-pares. O primeiro conjunto de experimentos demonstrou que a utilização de uma estratégia de *cache* de disco reduz a quantidade de transferências de dados entre domínios administrativos, independentemente da estratégia de substituição adotada (vide Seção 3.2). Em média, o serviço de *cache* de disco evitou a transferência de 40% do total de dados que deveria ser transferido caso o serviço de *cache* não existisse, nos cenários avaliados.

O segundo conjunto de experimentos de simulação (Seção 3.3) mostrou 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. Demonstrou-se, também, que o mecanismo realiza a alocação do espaço de forma bastante justa, proporcional à contribuição de cada nó ao sistema.

Os experimentos de escalabilidade (Seção 3.4) demonstraram que o mecanismo de incentivo à colaboração provê alocação justa e eficiente do espaço em disco doado para armazenamento, independente da quantidade de nós que compõem o sistema, marginalizando *free-riders* em momentos de contenção de recursos, mesmo que estes representem uma grande parcela dos nós.

A fim de avaliar-se o desempenho do mecanismo em uma grade P2P em produção, foi desenvolvido um protótipo do serviço proposto neste trabalho para o *middleware OurGrid*. Foi selecionada uma aplicação BoT *data-intensive* para a realização de experimentos de medição, denominada Meta-Dock [26]. A aplicação consiste na análise de ligantes que inibam a ação de proteínas que causam alguma enfermidade, onde cada tarefa representa a análise de um desses ligantes. Os experimentos demonstraram que houve uma redução do *makespan* em aproximadamente 40% quando comparado ao mesmo experimento realizado com *playpen*, e em aproximadamente 20% em relação a primeira execução com *storage*.

O protótipo implementado para o estudo de caso foi desenvolvido utilizando-se a versão 4.1 do *OurGrid* disponível em $http://www.ourgrid.org$. Atualmente, encontra-se em desenvolvimento uma versão do serviço de *cache* de disco para o *middleware OurGrid* baseada na implementação do protótipo descrita na Seção 4.2.

Dentre os possíveis trabalhos futuros, inclui-se o desenvolvimento de um mecanismo de compartilhamento de *caches*, de forma que elas se ajudem mutuamente, sendo o compartilhamento regido, também, por uma rede de favores. Outros direcionamentos para trabalhos futuros envolvem a implementação de um escalonador para o *middleware OurGrid* que utilize os benefícios providos por um serviço de *caching* de dados, como o proposto por Santos-Neto et al. [23]. Pretende-se, também, realizar um estudo da redução do tempo de execução de um conjunto mais abrangente de aplicações *data-intensive* que executam na grade P2P onde o serviço de *caching* é provido, como também a avaliação do desempenho do mecanismo de incentivo à colaboração na presença de *free-riders* nesta grade.

Bibliografia

[1] AutoDock. Disponível em http://autodock.scripps.edu/, Acessado em Janeiro 2010.

[2] NFS. Network File System. Disponível em http://www.rfc.org/specs/, Acessado em Janeiro 2010.

[3] XMPP. eXtensible Messaging and Presence Protocol Specification. Disponível em http://www.xmpp.org/specs/, Acessado em Janeiro 2010.

[4] ZINC. A free database of commercially-available compounds for virtual screening. Disponível em http://zinc.docking.org/, Acessado em Janeiro 2010.

[5] ADAR, E., AND HUBERMAN, B. A. Free Riding on Gnutella. *First Monday 5* (2000).

[6] ANDRADE, N., BRASILEIRO, F., CIRNE, W., AND MOWBRAY, M. Automatic grid assembly by promoting collaboration in peer-to-peer grids. *Journal of Parallel and Distributed Computing 67*, 8 (2007), 957–966.

[7] CASTRO, M., DRUSCHEL, P., GANESH, A., ROWSTRON, A., AND WALLACH, D. S. Security for structured peer-to-peer overlay networks. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI'02)* (December 2002).

[8] CIRNE, W., BRASILEIRO, F., ANDRADE, N., COSTA, L., ANDRADE, A., NOVAES, R., AND MOWBRAY, M. Labs of the World, Unite!!! *Journal of Grid Computing 4*, 3 (2006), 225–246.

[9] COX, L. P., AND NOBLE, B. D. Samsara: honor among thieves in peer-to-peer storage. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles* (New York, NY, USA, 2003), ACM, pp. 120–132.

[10] FELDMAN, M., AND CHUANG, J. Overcoming free-riding behavior in peer-to-peer systems. *SIGecom Exch. 5*, 4 (2005), 41–50.

[11] FIGUEIREDO, F., GAUDÊNCIO, M., EMMANUEL, T., MIRANDA, R., AND BRASILEIRO, F. Um portfólio de segurança para um sistema entre pares de livre entrada. In *VI Workshop de Computação em Grades e Aplicações* (2008), pp. 25–36.

[12] FOSTER, I., KESSELMAN, C., NICK, J. M., AND TUECKE, S. The Physiology of the Grid: An Open Grid Service Architecture for Distributed Systems Integration. *Open Service Infrastructure WG* (2002), 210–232.

[13] GOLLE, P., LEYTON-BROWN, K., MIRONOV, I., AND LILLIBRIDGE, M. Incentives for sharing in peer-to-peer networks. In *WELCOM '01: Proceedings of the Second International Workshop on Electronic Commerce* (London, UK, 2001), Springer-Verlag, pp. 75–87.

[14] HUGHES, D., COULSON, G., AND WALKERDINE, J. Free riding on gnutella revisited: the bell tolls? *Distributed Systems Online, IEEE 6*, 6 (June 2005).

[15] KARAKAYA, M., KORPEOGLU, I., AND ULUSOY, O. Free riding in peer-to-peer networks. *Internet Computing, IEEE 13*, 2 (March-April 2009), 92–98.

[16] LANDERS, M., ZHANG, H., AND TAN, K.-L. Peerstore: better performance by relaxing in peer-to-peer backup. In *Proceedings of Fourth International Conference on Peer-to-Peer Computing* (2004), pp. 72–79.

[17] MA, Y., AND WANG, D. An Approach to Fair Resource Sharing in Peer-to-Peer Systems. *Networking - ICN 2005 3421/2005* (2005), 643–652.

[18] MIRANDA, R. Providing security and a consistent failure detection semantic for asynchronous distributed objects. In *Proceedings of the* 4^{th} *Latin-American Symposium on Dependable Computing (LADC)* (João Pessoa, PB, Brasil, September 2009).

[19] NGAN, T., WALLACH, D., AND DRUSCHEL, P. Enforcing Fair Sharing of Peer-to-Peer Resources. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems* (2003).

[20] PARADA, J., PERDOMO, E., RIVAS, R., AND RUETTE, F. Cativic: A Grid Experience on Quantum Chemistry. In *Proceedings of the First EELA-2 Conference* (Bogotá, Colombia, 2009), Editorial CIEMAT, p. 10.

[21] PODLIPNIG, S., AND BÖSZÖRMENYI, L. A survey of web cache replacement strategies. *ACM Computing Surveys 35*, 4 (2003), 374–398.

[22] POLLACK, K. T., LONG, D. D. E., GOLDING, R. A., BECKER-SZENDY, R. A., AND REED, B. Quota enforcement for high-performance distributed storage systems. In *MSST '07: Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 72–86.

[23] SANTOS-NETO, E., CIRNE, W., BRASILEIRO, F., AND LIMA, A. Exploiting Replication and Data Reuse to Efficiently Schedule Data-Intensive Applications on Grids. *Job Scheduling Strategies for Parallel Processing 3277/2005* (2005), 210–232.

[24] SILVA, F., GAGLIARD, H., GALLO, E., MADOPE, M., NETO, V., PISA, I., AND ALVES, D. IntegraEPI: a Grid-based Epidemic Surveillance System. In *Proceedings of the HealthGrid Conference* (Switzerland, 2007).

[25] SOARES, P., OLIVEIRA, M., GUERRERO, D., AND BRASILEIRO, F. Solomon: Incentivando o Compartilhamento e Maior Disponibilidade em Sistemas de Armazenamento Entre-Pares. In *Anais do IV Workshop on Peer-to-Peer* (Recife, PE, Brasil, 2008), Sociedade Brasileira de Computação, pp. 1–12.

[26] VERLEYEN, J., HURTADO-RAMIREZ, J. M., AND MERINO-PEREZ, E. Meta-Dock, a gridification of a docking application. In *Proceedings of the First EELA-2 Conference* (Bogotá, Colombia, 2009), Editorial CIEMAT, p. 8.

Apêndice A

Resultados dos Experimentos

A.1 Avaliação do Impacto do Uso de *Cache* em Disco

Estratégia	**64**	ϵ	**128**	ϵ	**256**	ϵ	**512**	ϵ	**1.024**	ϵ
GDSF	17,4%	3,30%	15,0%	2,41%	17,1%	2,74%	23,2%	3,27%	26,8%	3,32%
LFU	17,4%	3,30%	15,1%	1,82%	17,2%	2,78%	22,6%	3,17%	27,6%	3,28%
LFU-DA	17,4%	3,30%	15,1%	1,81%	17,2%	2,78%	22,6%	3,17%	27,6%	3,28%
LRU	17,8%	3,43%	16,8%	1,20%	17,7%	3,02%	22,9%	3,25%	27,7%	3,29%

Tabela A.1: Taxa de acertos da *cache* em M*Bytes* para $r_{intra} = 25\%$

Estratégia	**64**	ϵ	**128**	ϵ	**256**	ϵ	**512**	ϵ	**1.024**	ϵ
GDSF	27,6%	3,83%	26,2%	2,47%	30,6%	2,70%	37,8%	3,13%	41,4%	3,30%
LFU	27,6%	3,84%	26,3%	2,05%	30,7%	2,75%	37,1%	3,04%	42,2%	3,30%
LFU-DA	27,6%	3,85%	26,3%	2,04%	30,7%	2,75%	37,1%	3,04%	42,2%	3,30%
LRU	28,5%	3,82%	29,0%	1,93%	31,3%	2,99%	37,3%	3,09%	42,2%	3,30%

Tabela A.2: Taxa de acertos da *cache* em M*Bytes* para $r_{intra} = 50\%$

Estratégia	**64**	ϵ	**128**	ϵ	**256**	ϵ	**512**	ϵ	**1.024**	ϵ
GDSF	40,6%	2,94%	42,3%	1,94%	51,7%	1,44%	60,2%	1,71%	63,1%	1,80%
LFU	40,7%	2,94%	44,0%	1,80%	51,8%	1,48%	59,7%	1,67%	63,7%	1,79%
LFU-DA	40,7%	2,94%	44,0%	1,79%	51,8%	1,48%	59,7%	1,67%	63,7%	1,79%
LRU	41,8%	2,83%	45,7%	1,64%	52,0%	1,52%	59,8%	1,67%	63,7%	1,79%

Tabela A.3: Taxa de acertos da *cache* em M*Bytes* para $r_{intra} = 75\%$

Estratégia	**64**	ϵ	**128**	ϵ	**256**	ϵ	**512**	ϵ	**1.024**	ϵ
GDSF	34,1%	2,92%	31,7%	3,48%	31,4%	4,39%	33,2%	4,04%	33,8%	4,85%
LFU	34,2%	2,93%	31,2%	3,77%	31,4%	4,39%	33,2%	4,10%	33,9%	4,80%
LFU-DA	34,2%	2,93%	31,1%	3,77%	31,4%	4,39%	33,2%	4,10%	33,9%	4,80%
LRU	35,0%	2,90%	32,9%	3,37%	31,8%	4,33%	33,3%	4,08%	34,0%	4,79%

Tabela A.4: Taxa de acertos da *cache* em M*Bytes* para $r_{inter} = 25\%$

Estratégia	**64**	ϵ	**128**	ϵ	**256**	ϵ	**512**	ϵ	**1.024**	ϵ
GDSF	26,9%	3,00%	28,0%	3,31%	30,1%	3,89%	36,8%	3,26%	40,1%	2,78%
LFU	26,9%	3,01%	27,3%	3,39%	30,2%	3,89%	36,1%	3,32%	40,9%	2,66%
LFU-DA	26,9%	3,01%	27,3%	3,40%	30,2%	3,89%	36,1%	3,32%	40,9%	2,66%
LRU	27,8%	3,02%	28,4%	3,05%	30,7%	3,90%	36,3%	3,31%	41,0%	2,65%

Tabela A.5: Taxa de acertos da *cache* em M*Bytes* para $r_{inter} = 50\%$

Estratégia	**64**	ϵ	**128**	ϵ	**256**	ϵ	**512**	ϵ	**1.024**	ϵ
GDSF	24,7%	4,83%	26,0%	4,07%	38,1%	3,69%	51,3%	2,77%	57,4%	2,06%
LFU	24,7%	4,84%	28,1%	3,99%	38,2%	3,71%	50,2%	2,86%	58,7%	1,84%
LFU-DA	24,7%	4,85%	28,1%	3,98%	38,2%	3,71%	50,2%	2,86%	58,7%	1,84%
LRU	25,4%	4,84%	30,3%	3,60%	38,6%	3,70%	50,4%	2,83%	58,8%	1,83%

Tabela A.6: Taxa de acertos da *cache* em M*Bytes* para $r_{inter} = 75\%$

Estratégia	**64**	ϵ	**128**	ϵ	**256**	ϵ	**512**	ϵ	**1.024**	ϵ
GDSF	22,4%	1,01%	26,9%	0,51%	32,5%	1,13%	37,6%	1,18%	43,1%	0,71%
LFU	22,2%	0,82%	26,8%	0,87%	30,0%	0,98%	33,7%	0,74%	37,6%	1,07%
LFU-DA	22,7%	1,34%	27,0%	1,06%	33,2%	1,02%	38,8%	0,64%	44,8%	1,25%
LRU	23,1%	1,02%	29,8%	0,93%	37,1%	1,29%	43,7%	1,11%	48,9%	1,03%

Tabela A.7: Taxa de acertos da *cache* em M*Bytes* para $r_{intra} = 75\%$ e $r_{inter} = 25\%$

Estratégia	**64**	ϵ	**128**	ϵ	**256**	ϵ	**512**	ϵ	**1.024**	ϵ
GDSF	14,7%	0,74%	22,9%	0,96%	28,4%	1,31%	43,2%	0,99%	52,9%	0,61%
LFU	14,9%	1,22%	23,4%	1,11%	30,7%	0,82%	46,4%	1,97%	58,0%	1,61%
LFU-DA	14,7%	1,08%	23,3%	1,47%	29,4%	0,91%	44,7%	1,26%	55,7%	1,42%
LRU	14,6%	1,13%	22,1%	0,59%	25,8%	1,15%	37,4%	0,72%	46,9%	1,39%

Tabela A.8: Taxa de acertos da *cache* em M*Bytes* para $r_{intra} = 25\%$ e $r_{inter} = 75\%$

A.2 Eficiência e Justiça

Turno	$N \cdot \beta = 25$	ϵ	$N \cdot \beta = 50$	ϵ	$N \cdot \beta = 75$	ϵ
1	15,1%	0,84%	18,3%	0,26%	25,5%	2,15%
2	7,7%	0,83%	9,1%	0,65%	8,3%	4,40%
3	5,0%	1,14%	6,9%	0,46%	4,7%	4,51%
4	3,4%	0,94%	6,3%	0,68%	3,5%	4,23%
5	2,6%	0,80%	5,0%	0,48%	2,7%	4,98%
6	1,9%	0,98%	4,6%	0,45%	2,0%	4,76%
7	1,7%	0,86%	5,2%	0,26%	1,7%	4,18%
8	1,4%	0,80%	4,2%	0,50%	1,7%	4,27%
9	1,5%	0,59%	4,4%	0,41%	1,6%	4,89%
10	1,0%	0,60%	4,4%	0,31%	1,3%	4,65%
11	1,0%	0,97%	3,8%	0,45%	1,2%	4,54%
12	0,9%	0,75%	3,9%	0,48%	0,8%	4,83%
13	1,1%	0,73%	3,5%	0,38%	0,7%	4,94%
14	0,9%	0,80%	3,6%	0,34%	0,7%	4,66%
15	0,6%	0,77%	3,4%	0,49%	0,9%	4,26%
16	0,8%	0,53%	3,6%	0,49%	0,6%	4,43%
17	0,8%	0,90%	3,6%	0,28%	0,7%	4,27%
18	0,7%	0,73%	3,7%	0,44%	0,7%	4,62%
19	0,8%	0,43%	3,4%	0,38%	0,5%	4,00%
20	0,4%	0,72%	3,6%	0,32%	0,7%	3,59%
21	0,6%	0,99%	3,6%	0,55%	0,6%	3,01%
22	0,5%	0,61%	3,4%	0,46%	0,6%	4,11%
23	0,7%	0,74%	3,3%	0,38%	0,6%	4,78%
24	0,4%	0,53%	3,2%	0,30%	0,6%	4,94%
25	0,6%	0,68%	3,5%	0,33%	0,5%	4,32%
26	0,3%	0,83%	3,6%	0,23%	0,4%	2,92%
27	0,5%	0,89%	3,3%	0,37%	0,5%	4,81%
28	0,5%	0,79%	3,3%	0,34%	0,4%	1,18%
29	0,6%	0,93%	3,2%	0,45%	0,3%	4,34%
30	0,3%	0,88%	3,3%	0,45%	0,5%	2,46%
31	0,4%	0,88%	3,3%	0,22%	0,3%	4,20%
32	0,2%	0,74%	3,2%	0,30%	0,3%	4,06%
33	0,3%	0,84%	3,1%	0,20%	0,4%	3,36%
34	0,3%	0,72%	3,1%	0,42%	0,4%	4,63%
35	0,4%	0,80%	3,3%	0,44%	0,5%	4,23%
36	0,3%	0,64%	3,4%	0,44%	0,4%	4,09%
37	0,5%	0,62%	3,4%	0,28%	0,4%	4,50%
38	0,3%	0,79%	3,0%	0,22%	0,3%	2,71%
39	0,2%	0,72%	3,2%	0,42%	0,3%	4,34%
40	0,4%	0,83%	3,2%	0,20%	0,4%	4,83%

Tabela A.9: Nível de satisfação médio para *free-riders* - alta contenção com $f = 50\%$ e diferentes valores de $N \cdot \beta$

Turno	$f = 25\%$	ϵ	$f = 50\%$	ϵ	$f = 75\%$	ϵ
1	16,5%	3,81 %	18,3%	4,20 %	29,7%	0,99 %
2	8,4%	4,52 %	9,1%	4,10 %	6,0%	4,83 %
3	6,3%	4,20 %	6,9%	4,96 %	4,0%	1,26 %
4	5,2%	2,28 %	6,3%	4,51 %	3,0%	4,17 %
5	4,7%	3,66 %	5,0%	4,77 %	2,4%	1,59 %
6	4,4%	4,33 %	4,6%	3,11 %	2,0%	4,52 %
7	4,0%	4,48 %	5,2%	4,62 %	1,9%	2,74 %
8	3,7%	4,01 %	4,2%	4,37 %	1,4%	1,50 %
9	3,9%	4,77 %	4,4%	4,665 %	1,3%	4,42 %
10	3,4%	3,13 %	4,4%	4,50 %	1,5%	4,97 %
11	3,6%	3,44 %	3,8%	4,30 %	0,8%	2,46 %
12	3,0%	4,42 %	3,9%	4,78 %	0,9%	1,81 %
13	2,5%	4,72 %	3,5%	4,78 %	0,7%	2,59 %
14	2,6%	4,79 %	3,6%	4,67 %	0,8%	4,74 %
15	2,9%	4,46 %	3,4%	4,89 %	0,6%	3,05 %
16	2,8%	4,08 %	3,6%	4,58 %	0,6%	4,27 %
17	2,5%	4,14 %	3,6%	4,70 %	0,7%	4,64 %
18	2,6%	4,72 %	3,7%	4,28 %	0,6%	4,52 %
19	2,4%	3,86 %	3,4%	4,66 %	0,7%	4,65 %
20	2,5%	4,63 %	3,6%	4,99 %	0,6%	4,81 %
21	2,8%	4,29 %	3,6%	4,43 %	0,6%	2,89 %
22	2,8%	4,93 %	3,4%	4,73 %	0,5%	4,90 %
23	2,4%	4,32 %	3,3%	4,90 %	0,5%	4,18 %
24	2,4%	4,71 %	3,2%	4,60 %	0,4%	1,60 %
25	2,5%	2,46 %	3,5%	4,65 %	0,5%	3,78 %
26	2,5%	2,78 %	3,6%	4,33 %	0,4%	3,52 %
27	2,4%	3,58 %	3,3%	4,13 %	0,5%	1,48 %
28	2,2%	1,92 %	3,3%	4,95 %	0,3%	4,92 %
29	2,4%	4,81 %	3,2%	4,21 %	0,3%	4,41 %
30	2,2%	4,02 %	3,3%	4,31 %	0,5%	4,41 %
31	2,1%	2,64 %	3,3%	4,26 %	0,4%	4,37 %
32	2,2%	4,53 %	3,2%	4,01 %	0,3%	4,63 %
33	2,3%	4,61 %	3,1%	4,57 %	0,2%	4,76 %
34	2,4%	1,24 %	3,1%	4,25 %	0,4%	1,58 %
35	2,1%	4,07 %	3,3%	4,41 %	0,4%	4,33 %
36	2,3%	2,59 %	3,4%	4,10 %	0,3%	4,92 %
37	2,1%	4,67 %	3,4%	4,45 %	0,4%	4,32 %
38	2,1%	4,69 %	3,0%	4,77 %	0,3%	3,28 %
39	2,1%	4,57 %	3,2%	4,36 %	0,3%	4,47 %
40	2,1%	4,63 %	3,2%	4,89 %	0,3%	1,55 %

Tabela A.10: Nível de satisfação médio para *free-riders* - alta contenção com $N \cdot \beta = 50$ e diferentes valores de f

Turno	$N \cdot \beta = 25$	ϵ	$N \cdot \beta = 50$	ϵ	$N \cdot \beta = 75$	ϵ
1	100,0%	0,00 %	100,0%	0,00 %	100,0%	0,00 %
2	100,0%	0,00 %	93,6%	1,07 %	98,2%	1,18 %
3	49,6%	4,81 %	19,1%	1,96 %	26,5%	4,64 %
4	16,9%	4,16 %	13,0%	4,24 %	15,7%	4,46 %
5	11,5%	4,26 %	10,4%	3,19 %	10,7%	2,95 %
6	9,4%	4,58 %	8,3%	4,90 %	8,5%	3,79 %
7	7,4%	1,48 %	7,5%	4,35 %	6,9%	4,84 %
8	6,2%	4,32 %	6,3%	4,52 %	6,0%	4,94 %
9	5,4%	3,72 %	5,1%	4,21 %	5,9%	4,22 %
10	4,3%	4,13 %	5,1%	1,83 %	5,2%	4,10 %
11	4,1%	4,60 %	4,2%	4,40 %	4,4%	4,49 %
12	4,0%	4,60 %	4,0%	2,24 %	3,8%	4,84 %
13	3,2%	4,20 %	3,2%	2,98 %	3,3%	4,91 %
14	3,3%	1,17 %	3,5%	1,00 %	3,5%	1,33 %
15	2,5%	4,49 %	2,9%	4,86 %	3,4%	4,85 %
16	3,0%	4,74 %	2,9%	4,89 %	2,8%	4,84 %
17	2,7%	4,89 %	2,6%	4,58 %	2,9%	4,97 %
18	2,6%	4,20 %	2,6%	1,61 %	2,8%	2,17 %
19	2,5%	4,42 %	2,7%	4,36 %	2,6%	3,59 %
20	2,0%	4,41 %	2,5%	4,66 %	2,4%	2,97 %
21	2,2%	4,09 %	2,6%	1,25 %	2,3%	4,67 %
22	1,9%	3,64 %	2,3%	1,56 %	2,1%	4,95 %
23	1,9%	2,45 %	1,9%	3,68 %	2,2%	4,57 %
24	1,8%	4,68 %	2,2%	4,88 %	1,9%	2,39 %
25	2,1%	3,54 %	1,7%	4,70 %	2,0%	4,41 %
26	1,3%	1,07 %	2,0%	4,01 %	1,8%	4,85 %
27	1,9%	4,40 %	1,8%	4,93 %	1,9%	4,39 %
28	1,5%	3,73 %	1,5%	2,58 %	1,6%	3,40 %
29	1,6%	2,75 %	1,4%	2,50 %	1,8%	3,58 %
30	1,6%	4,83 %	1,7%	1,41 %	1,7%	4,81 %
31	1,3%	4,62 %	1,6%	4,73 %	1,6%	4,71 %
32	1,4%	1,59 %	1,3%	3,91 %	1,5%	2,56 %
33	1,3%	2,89 %	1,0%	4,20 %	1,6%	2,04 %
34	1,2%	4,23 %	1,1%	1,88 %	1,4%	4,39 %
35	1,5%	4,13 %	1,1%	4,68 %	1,5%	4,42 %
36	1,3%	4,74 %	0,9%	1,23 %	1,4%	4,14 %
37	1,3%	4,07 %	0,9%	2,10 %	1,3%	1,57 %
38	1,1%	4,10 %	0,9%	1,33 %	1,2%	4,00 %
39	1,3%	3,64 %	0,9%	4,35 %	1,3%	4,30 %
40	1,2%	4,12 %	0,9%	4,74 %	1,4%	4,04 %

Tabela A.11: Nível de satisfação médio para *free-riders* - baixa contenção com $f = 50\%$ e diferentes valores de $N \cdot \beta$

Turno	$f = 25\%$	ϵ	$f = 50\%$	ϵ	$f = 75\%$	ϵ
1	100,0%	0,00 %	100,0%	0,00 %	84,0%	3,13 %
2	43,7%	3,95 %	93,6%	1,07 %	57,6%	3,60 %
3	22,0%	4,64 %	19,1%	1,96 %	18,5%	0,49 %
4	16,9%	4,03 %	13,0%	4,24 %	12,5%	4,31 %
5	13,9%	4,83 %	10,4%	3,19 %	9,0%	4,19 %
6	12,4%	4,51 %	8,3%	4,90 %	7,7%	4,75 %
7	12,0%	4,06 %	7,5%	4,35 %	6,6%	4,07 %
8	9,9%	4,71 %	6,3%	4,52 %	5,7%	4,14 %
9	10,1%	4,73 %	5,1%	4,21 %	4,8%	4,26 %
10	9,2%	4,48 %	5,1%	1,83 %	4,7%	4,44 %
11	8,3%	4,83 %	4,2%	4,40 %	4,0%	4,01 %
12	7,7%	4,52 %	4,0%	2,24 %	3,6%	0,88 %
13	6,9%	4,78 %	3,2%	2,98 %	3,3%	4,87 %
14	7,3%	4,24 %	3,5%	1,00 %	3,1%	1,24 %
15	6,4%	4,53 %	2,9%	4,86 %	2,9%	0,85 %
16	7,2%	1,22 %	2,9%	4,89 %	2,7%	4,16 %
17	6,4%	4,11 %	2,6%	4,58 %	2,9%	4,15 %
18	6,8%	1,75 %	2,6%	1,61 %	2,3%	4,18 %
19	6,3%	3,22 %	2,7%	4,36 %	2,5%	2,70 %
20	6,1%	4,60 %	2,5%	4,66 %	2,5%	1,64 %
21	6,7%	4,92 %	2,6%	1,25 %	2,2%	0,79 %
22	5,8%	4,62 %	2,3%	1,56 %	2,1%	4,62 %
23	5,7%	4,43 %	1,9%	3,68 %	1,8%	3,20 %
24	5,5%	3,86 %	2,2%	4,88 %	1,9%	4,97 %
25	5,6%	2,91 %	1,7%	4,70 %	1,9%	4,41 %
26	5,5%	4,89 %	2,0%	4,01 %	1,6%	4,42 %
27	5,4%	4,06 %	1,8%	4,93 %	1,7%	3,63 %
28	4,8%	4,27 %	1,5%	2,58 %	1,6%	1,21 %
29	5,2%	4,50 %	1,4%	2,50 %	1,5%	4,64 %
30	4,8%	4,56 %	1,7%	1,41 %	1,7%	4,97 %
31	5,3%	4,15 %	1,6%	4,73 %	1,5%	1,51 %
32	5,0%	4,27 %	1,3%	3,91 %	1,3%	2,61 %
33	5,2%	3,40 %	1,0%	4,20 %	1,3%	4,51 %
34	4,7%	3,14 %	1,1%	1,88 %	1,3%	3,51 %
35	5,5%	4,37 %	1,1%	4,68 %	1,6%	2,97 %
36	5,1%	4,31 %	0,9%	1,23 %	1,4%	4,81 %
37	4,9%	1,68 %	0,9%	2,10 %	1,5%	4,68 %
38	4,4%	4,65 %	0,9%	1,33 %	0,9%	2,67 %
39	4,7%	2,61 %	0,9%	4,35 %	0,7%	4,65 %
40	4,5%	3,62 %	0,9%	4,74 %	0,7%	2,55 %

Tabela A.12: Nível de satisfação médio para *free-riders* - baixa contenção com $n = 50$ e diferentes valores de f

Turno	$N\beta = 25$	ϵ	$N(1-\beta) = 75$	ϵ	$N\beta = 50$	ϵ	$N(1-\beta) = 50$	ϵ
1	85,9%	0,19 %	72,6%	0,98 %	76,9%	0,87 %	68,2%	3,12 %
2	72,0%	1,87 %	53,7%	2,86 %	55,5%	1,55 %	53,2%	3,72 %
3	69,4%	2,93 %	47,3%	2,74 %	49,4%	1,60 %	47,2%	3,30 %
4	68,1%	3,48 %	44,0%	2,74 %	45,7%	1,28 %	43,5%	3,61 %
5	67,3%	3,71 %	41,1%	2,74 %	42,2%	1,18 %	40,2%	3,74 %
6	66,7%	3,88 %	39,9%	3,29 %	38,2%	1,43 %	38,9%	4,40 %
7	66,1%	4,32 %	39,3%	2,96 %	37,7%	1,39 %	38,1%	4,06 %
8	66,1%	4,16 %	38,4%	3,02 %	36,5%	1,40 %	36,7%	3,93 %
9	65,4%	4,50 %	37,8%	2,95 %	35,8%	1,27 %	36,2%	4,18 %
10	66,0%	4,20 %	37,5%	3,06 %	35,7%	1,22 %	37,5%	4,14 %
11	65,2%	4,48 %	37,2%	3,30 %	35,3%	1,46 %	35,4%	4,33 %
12	65,7%	4,29 %	37,3%	3,26 %	35,4%	1,48 %	35,8%	4,27 %
13	64,7%	4,88 %	36,4%	3,32 %	34,3%	1,59 %	34,9%	4,32 %
14	64,7%	4,96 %	36,5%	3,13 %	34,6%	1,31 %	34,9%	4,35 %
15	65,0%	4,63 %	35,6%	3,65 %	32,7%	1,35 %	34,0%	4,30 %
16	64,4%	4,12 %	35,4%	3,44 %	33,3%	1,61 %	33,5%	4,64 %
17	65,0%	4,75 %	35,6%	3,49 %	33,6%	1,77 %	33,8%	4,48 %
18	64,8%	4,86 %	35,3%	3,67 %	36,2%	1,60 %	32,6%	4,07 %
19	64,6%	4,93 %	35,1%	3,81 %	34,0%	1,71 %	33,0%	4,24 %
20	64,7%	4,90 %	35,3%	3,35 %	34,2%	1,50 %	33,5%	4,70 %
21	64,7%	4,86 %	34,8%	3,51 %	33,7%	1,56 %	33,0%	4,58 %
22	64,7%	4,89 %	34,7%	3,54 %	32,5%	1,67 %	32,8%	4,64 %
23	65,0%	4,59 %	35,1%	3,48 %	33,0%	1,27 %	31,1%	4,93 %
24	64,2%	4,11 %	34,6%	3,87 %	32,6%	1,54 %	31,5%	4,75 %
25	64,6%	4,93 %	34,9%	3,62 %	32,7%	1,68 %	32,5%	4,04 %
26	64,5%	4,96 %	34,6%	3,80 %	32,4%	1,60 %	33,4%	4,51 %
27	64,3%	4,09 %	34,5%	3,89 %	32,3%	1,77 %	33,8%	4,30 %
28	64,5%	4,06 %	34,3%	4,03 %	32,1%	1,89 %	32,3%	4,56 %
29	64,7%	4,83 %	34,4%	3,83 %	32,2%	1,74 %	32,5%	4,05 %
30	64,4%	4,06 %	34,4%	4,03 %	32,3%	2,02 %	32,4%	4,19 %
31	64,6%	4,97 %	34,3%	3,80 %	32,1%	1,57 %	31,9%	4,61 %
32	64,0%	4,28 %	34,1%	3,86 %	32,1%	1,95 %	32,1%	4,94 %
33	64,6%	4,06 %	34,2%	3,71 %	32,0%	1,62 %	31,8%	4,24 %
34	64,4%	4,03 %	34,2%	3,77 %	32,1%	1,44 %	32,1%	4,52 %
35	64,1%	4,17 %	34,0%	3,97 %	31,9%	1,69 %	31,7%	4,82 %
36	64,4%	4,90 %	34,0%	3,60 %	31,7%	1,45 %	31,7%	4,02 %
37	64,3%	4,13 %	33,9%	3,78 %	31,6%	1,73 %	31,8%	4,15 %
38	64,1%	4,28 %	34,2%	3,85 %	32,0%	1,82 %	32,2%	4,20 %
39	64,2%	4,08 %	34,0%	3,71 %	31,7%	1,52 %	32,0%	4,26 %
40	64,1%	4,28 %	33,6%	4,05 %	31,4%	1,92 %	31,3%	4,59 %

Tabela A.13: Nível de satisfação médio para colaboradores (média contenção), $N \cdot \beta = 25$ e $N \cdot \beta = 50$

Turno	$N \cdot \beta = 75$	ϵ	$N(1-\beta) = 25$	ϵ
1	86,8%	0,23 %	71,2%	1,76 %
2	72,8%	1,50 %	53,0%	3,63 %
3	69,9%	2,68 %	46,1%	3,54 %
4	68,6%	3,11 %	42,6%	3,70 %
5	67,5%	3,54 %	39,4%	3,83 %
6	66,8%	3,78 %	38,2%	4,36 %
7	66,2%	4,15 %	37,7%	3,99 %
8	66,3%	4,09 %	36,5%	4,05 %
9	66,0%	4,16 %	35,8%	4,25 %
10	65,8%	4,27 %	35,7%	4,19 %
11	65,6%	4,44 %	35,3%	4,50 %
12	65,6%	4,31 %	35,4%	4,45 %
13	65,0%	4,65 %	34,3%	4,58 %
14	65,0%	4,80 %	34,6%	4,26 %
15	64,8%	4,73 %	33,7%	4,96 %
16	64,7%	4,86 %	33,3%	4,75 %
17	65,0%	4,74 %	33,6%	4,62 %
18	64,9%	4,76 %	33,2%	4,99 %
19	64,8%	4,85 %	33,0%	4,14 %
20	64,8%	4,76 %	33,2%	4,76 %
21	64,6%	4,91 %	32,7%	4,78 %
22	64,9%	4,78 %	32,5%	4,95 %
23	65,0%	4,70 %	33,0%	4,77 %
24	64,2%	4,10 %	32,6%	4,20 %
25	64,8%	4,79 %	32,7%	4,06 %
26	64,7%	4,88 %	32,4%	4,23 %
27	64,7%	4,93 %	32,3%	4,27 %
28	64,6%	4,99 %	32,1%	4,40 %
29	64,6%	4,84 %	32,2%	4,26 %
30	64,4%	4,02 %	32,3%	4,40 %
31	64,4%	4,95 %	32,1%	4,30 %
32	64,0%	4,22 %	32,1%	4,22 %
33	64,4%	4,06 %	32,0%	4,11 %
34	64,4%	4,99 %	32,1%	4,21 %
35	64,2%	4,14 %	31,9%	4,39 %
36	64,5%	4,88 %	31,7%	4,03 %
37	64,6%	4,87 %	31,6%	4,36 %
38	64,3%	4,10 %	32,0%	4,31 %
39	64,4%	4,89 %	31,7%	4,26 %
40	64,1%	4,17 %	31,4%	4,59 %

Tabela A.14: Nível de satisfação médio para colaboradores (média contenção), $N \cdot \beta = 75$

Turno	$N\beta = 25$	ϵ	$N(1-\beta) = 75$	ϵ	$N\beta = 50$	ϵ	$N(1-\beta) = 50$	ϵ
1	55,7%	1,21 %	53,1%	1,74 %	61,7%	1,07 %	53,4%	3,56 %
2	42,1%	1,78 %	33,2%	1,83 %	39,3%	1,23 %	38,2%	4,22 %
3	39,3%	2,93 %	27,7%	1,34 %	34,6%	1,76 %	31,3%	3,32 %
4	38,2%	3,83 %	24,3%	1,74 %	30,6%	1,43 %	28,4%	3,12 %
5	37,4%	3,76 %	21,0%	1,41 %	27,3%	1,29 %	25,1%	3,96 %
6	36,6%	2,32 %	20,1%	1,92 %	23,1%	1,12 %	23,9%	4,20 %
7	36,0%	4,32 %	19,3%	1,69 %	22,5%	1,78 %	23,2%	4,01 %
8	36,1%	4,16 %	18,3%	1,02 %	21,3%	1,54 %	21,8%	3,93 %
9	35,4%	4,50 %	17,9%	1,95 %	20,9%	1,17 %	21,1%	4,07 %
10	36,0%	4,20 %	17,3%	1,03 %	20,7%	1,02 %	22,5%	4,53 %
11	35,2%	4,48 %	17,1%	1,32 %	20,3%	1,37 %	20,4%	3,33 %
12	35,7%	4,29 %	17,4%	2,26 %	20,4%	1,23 %	20,8%	3,21 %
13	34,7%	4,88 %	16,4%	1,32 %	19,3%	1,86 %	19,9%	4,32 %
14	34,7%	4,96 %	16,5%	2,31 %	19,6%	1,11 %	19,8%	4,35 %
15	35,0%	4,63 %	15,6%	1,65 %	17,6%	1,45 %	19,1%	3,30 %
16	34,4%	2,12 %	15,4%	2,44 %	18,3%	1,32 %	18,3%	4,64 %
17	35,0%	4,75 %	15,6%	1,19 %	18,6%	1,99 %	18,6%	4,48 %
18	34,8%	4,86 %	15,3%	3,57 %	18,1%	1,54 %	17,5%	3,07 %
19	34,6%	4,93 %	15,1%	3,87 %	19,0%	1,70 %	18,0%	3,24 %
20	34,7%	4,90 %	15,3%	3,35 %	19,2%	1,51 %	18,6%	3,70 %
21	34,7%	4,86 %	14,8%	3,54 %	18,6%	1,57 %	18,2%	4,85 %
22	34,7%	4,89 %	14,7%	3,34 %	18,3%	1,66 %	17,6%	3,64 %
23	35,0%	4,59 %	15,1%	3,18 %	18,0%	1,24 %	16,0%	4,92 %
24	34,2%	3,11 %	14,6%	3,27 %	17,6%	1,55 %	16,3%	3,71 %
25	34,8%	4,93 %	14,8%	3,62 %	17,7%	1,60 %	17,4%	3,04 %
26	34,5%	4,96 %	14,6%	3,10 %	17,4%	1,76 %	18,5%	4,51 %
27	34,3%	3,09 %	14,5%	3,89 %	17,3%	1,79 %	18,6%	4,31 %
28	34,4%	3,06 %	14,3%	4,03 %	17,1%	1,98 %	17,2%	3,23 %
29	34,7%	3,83 %	14,4%	3,83 %	17,3%	1,47 %	17,6%	3,67 %
30	34,4%	3,06 %	14,4%	4,03 %	17,5%	1,21 %	17,3%	3,10 %
31	34,6%	3,97 %	14,3%	3,80 %	17,1%	2,71 %	16,7%	3,95 %
32	34,0%	3,28 %	14,1%	3,86 %	17,2%	1,58 %	17,2%	4,43 %
33	34,6%	4,06 %	14,2%	2,71 %	17,2%	1,24 %	16,8%	3,32 %
34	34,4%	3,03 %	14,3%	3,77 %	17,1%	1,31 %	17,1%	3,64 %
35	34,2%	4,17 %	14,0%	3,97 %	16,9%	1,96 %	16,6%	3,24 %
36	34,5%	2,90 %	14,1%	1,60 %	16,8%	1,52 %	16,5%	3,18 %
37	34,3%	3,13 %	13,7%	1,78 %	16,4%	1,36 %	16,9%	3,02 %
38	34,5%	4,28 %	14,3%	1,95 %	17,1%	1,24 %	17,3%	3,24 %
39	34,6%	3,08 %	14,1%	1,34 %	16,6%	1,23 %	17,1%	3,95 %
40	34,0%	4,28 %	13,4%	2,56 %	16,2%	1,63 %	16,5%	3,62 %

Tabela A.15: Nível de satisfação médio para colaboradores (alta contenção), $N \cdot \beta = 25$ e $N \cdot \beta = 50$

Turno	$N \cdot \beta = 75$	ϵ	$N(1-\beta) = 25$	ϵ
1	61,8%	7,23 %	61,2%	11,76 %
2	47,8%	11,50 %	43,0%	13,63 %
3	44,9%	12,68 %	36,1%	13,54 %
4	43,6%	13,11 %	32,6%	13,70 %
5	42,5%	13,54 %	29,4%	13,83 %
6	41,8%	13,78 %	28,2%	14,36 %
7	41,2%	14,15 %	27,7%	13,99 %
8	41,3%	14,09 %	26,5%	14,05 %
9	41,0%	14,16 %	25,8%	14,25 %
10	40,8%	14,27 %	25,7%	14,19 %
11	40,6%	14,44 %	25,3%	14,50 %
12	40,6%	14,31 %	25,4%	14,45 %
13	40,0%	14,65 %	24,3%	14,58 %
14	40,0%	14,80 %	24,6%	14,26 %
15	39,8%	14,73 %	23,7%	14,96 %
16	39,7%	14,86 %	23,3%	14,75 %
17	40,0%	14,74 %	23,6%	14,62 %
18	39,9%	14,76 %	23,2%	14,99 %
19	39,8%	14,85 %	23,0%	15,14 %
20	39,8%	14,76 %	23,2%	14,76 %
21	39,6%	14,91 %	22,7%	14,78 %
22	39,9%	14,78 %	22,5%	14,95 %
23	40,0%	14,70 %	23,0%	14,77 %
24	39,2%	15,10 %	22,6%	15,20 %
25	39,8%	14,79 %	22,7%	15,06 %
26	39,7%	14,88 %	22,4%	15,23 %
27	39,7%	14,93 %	22,3%	15,27 %
28	39,6%	14,99 %	22,1%	15,40 %
29	39,6%	14,84 %	22,2%	15,26 %
30	39,4%	15,02 %	22,3%	15,40 %
31	39,4%	14,95 %	22,1%	15,30 %
32	39,0%	15,22 %	22,1%	15,22 %
33	39,4%	15,06 %	22,0%	15,11 %
34	39,4%	14,99 %	22,1%	15,21 %
35	39,2%	15,14 %	21,9%	15,39 %
36	39,5%	14,88 %	21,7%	15,03 %
37	39,6%	14,87 %	21,6%	15,36 %
38	39,3%	15,10 %	22,0%	15,31 %
39	39,4%	14,89 %	21,7%	15,26 %
40	39,1%	15,17 %	21,4%	15,59 %

Tabela A.16: Nível de satisfação médio para colaboradores (alta contenção), $N \cdot \beta = 75$

A.3 Escalabilidade

Turno	$c = 25\%$	ϵ	$f = 75\%$	ϵ	$c = 50\%$	ϵ	$f = 50\%$	ϵ
1	99,9%	0,00 %	45,4%	2,86 %	100,0%	0,00 %	100,0%	0,00 %
2	66,7%	4,86 %	12,4%	3,99 %	100,0%	0,00 %	100,0%	0,00 %
3	54,8%	1,11 %	8,9%	4,48 %	100,0%	0,00 %	100,0%	0,00 %
4	55,7%	0,07 %	9,4%	4,21 %	100,0%	0,00 %	100,0%	0,00 %
5	55,9%	1,07 %	10,2%	4,45 %	100,0%	0,00 %	100,0%	0,00 %
6	55,5%	1,58 %	10,7%	4,23 %	100,0%	0,00 %	100,0%	0,00 %
7	54,4%	0,74 %	10,1%	4,71 %	100,0%	0,00 %	100,0%	0,00 %
8	54,1%	0,26 %	10,1%	0,26 %	100,0%	0,00 %	100,0%	0,00 %
9	55,0%	0,35 %	9,7%	4,45 %	100,0%	0,00 %	100,0%	0,00 %
10	53,9%	0,80 %	9,5%	2,77 %	99,9%	0,01 %	71,4%	4,17 %
11	54,1%	1,38 %	9,6%	4,20 %	91,2%	3,56 %	25,1%	4,72 %
12	53,1%	1,42 %	9,0%	1,58 %	70,7%	0,68 %	19,0%	4,32 %
13	52,7%	2,57 %	9,0%	4,41 %	68,8%	1,84 %	16,2%	1,20 %
14	53,2%	4,13 %	9,1%	0,22 %	68,0%	1,07 %	14,1%	2,67 %
15	52,7%	1,57 %	9,3%	2,51 %	65,5%	0,61 %	14,3%	4,99 %
16	53,0%	0,20 %	9,2%	4,85 %	65,9%	0,99 %	13,5%	0,79 %
17	52,8%	1,66 %	8,6%	4,26 %	64,9%	1,39 %	13,8%	3,44 %
18	53,1%	1,41 %	8,6%	4,83 %	67,3%	1,67 %	12,4%	3,93 %
19	52,3%	0,75 %	7,7%	2,38 %	65,8%	2,39 %	12,2%	3,17 %
20	53,0%	1,65 %	9,2%	4,33 %	64,0%	0,93 %	11,9%	2,03 %
21	52,8%	0,54 %	8,6%	4,27 %	64,5%	3,13 %	11,8%	4,60 %
22	52,6%	0,87 %	7,7%	4,32 %	63,6%	2,69 %	10,9%	4,54 %
23	52,9%	0,63 %	7,2%	4,35 %	63,5%	3,04 %	10,3%	0,12 %
24	53,1%	1,79 %	7,5%	4,30 %	62,5%	1,55 %	10,9%	4,98 %
25	53,0%	2,14 %	7,4%	4,64 %	62,1%	1,89 %	11,0%	4,12 %
26	52,8%	3,45 %	7,3%	1,38 %	62,2%	1,40 %	10,1%	0,24 %
27	52,7%	2,71 %	7,1%	1,42 %	61,6%	0,45 %	9,9%	1,20 %
28	52,5%	1,64 %	6,5%	2,57 %	62,0%	2,35 %	9,3%	4,37 %
29	52,1%	0,97 %	7,3%	4,13 %	61,5%	1,21 %	8,3%	3,26 %
30	52,3%	0,32 %	6,7%	4,02 %	60,0%	0,26 %	8,6%	2,48 %
31	51,8%	1,49 %	6,3%	4,15 %	58,5%	0,35 %	7,9%	3,72 %
32	51,5%	1,93 %	6,5%	4,20 %	61,2%	0,80 %	8,5%	3,51 %
33	51,7%	2,11 %	6,4%	4,26 %	61,0%	1,38 %	9,2%	2,74 %
34	51,2%	1,43 %	6,3%	3,42 %	61,2%	1,42 %	8,6%	1,97 %
35	52,9%	0,94 %	7,0%	4,89 %	60,7%	2,57 %	7,7%	4,23 %
36	51,4%	1,67 %	6,7%	4,23 %	60,1%	1,11 %	7,2%	4,63 %
37	51,3%	3,67 %	6,5%	3,79 %	59,6%	0,87 %	7,5%	3,51 %
38	51,7%	1,23 %	6,2%	4,33 %	59,4%	0,49 %	7,1%	4,18 %
39	51,5%	1,00 %	6,0%	4,26 %	59,8%	1,36 %	6,7%	4,89 %
40	51,1%	0,57 %	6,7%	4,76 %	59,3%	0,65 %	7,0%	3,21 %

Tabela A.17: Nível de satisfação médio para colaboradores (c) e *free-riders* (f) para 1.000 nós, $c = 25\%$ e $c = 50\%$

Turno	$c = 75\%$	ϵ	$f = 25\%$	ϵ
1	100,0%	0,00 %	100,0%	0,00 %
2	100,0%	0,00 %	100,0%	0,00 %
3	100,0%	0,00 %	100,0%	0,00 %
4	100,0%	0,00 %	100,0%	0,00 %
5	100,0%	0,00 %	100,0%	0,00 %
6	100,0%	0,00 %	100,0%	0,00 %
7	100,0%	0,00 %	100,0%	0,00 %
8	100,0%	0,00 %	100,0%	0,00 %
9	100,0%	0,00 %	100,0%	0,00 %
10	100,0%	0,00 %	100,0%	0,00 %
11	100,0%	0,00 %	100,0%	0,00 %
12	100,0%	0,00 %	100,0%	0,00 %
13	100,0%	0,00 %	100,0%	0,00 %
14	90,0%	0,55 %	24,7%	4,33 %
15	75,8%	0,89 %	20,4%	4,55 %
16	74,4%	2,23 %	21,7%	4,78 %
17	73,4%	0,88 %	23,2%	4,97 %
18	72,1%	1,30 %	21,3%	4,31 %
19	71,7%	1,66 %	19,6%	1,63 %
20	72,1%	0,54 %	20,4%	4,73 %
21	71,9%	0,68 %	20,5%	4,15 %
22	69,7%	0,15 %	19,2%	1,58 %
23	69,3%	0,24 %	18,4%	3,24 %
24	69,1%	0,96 %	18,2%	2,78 %
25	68,3%	1,05 %	18,5%	3,63 %
26	68,7%	0,86 %	17,7%	4,12 %
27	68,4%	0,43 %	17,4%	4,09 %
28	67,9%	1,25 %	17,8%	4,37 %
29	67,3%	0,92 %	15,8%	3,69 %
30	67,0%	0,31 %	16,8%	3,23 %
31	66,6%	0,67 %	16,7%	3,48 %
32	67,1%	1,32 %	16,5%	2,41 %
33	65,2%	0,74 %	16,2%	4,89 %
34	66,0%	0,96 %	15,7%	1,67 %
35	65,8%	0,67 %	15,6%	2,98 %
36	65,3%	0,12 %	15,2%	3,11 %
37	65,0%	0,03 %	16,1%	4,03 %
38	64,6%	0,54 %	15,3%	4,45 %
39	64,8%	1,47 %	15,1%	3,76 %
40	65,2%	0,77 %	14,4%	2,34 %

Tabela A.18: Nível de satisfação médio para colaboradores (c) e *free-riders* (f) para 1.000 nós, $c = 75\%$

Turno	$c = 25\%$	ϵ	$f = 75\%$	ϵ	$c = 50\%$	ϵ	$f = 50\%$	ϵ
1	100,0%	0,00 %	35,9%	1,42 %	100,0%	0,00 %	100,0%	0,00 %
2	85,2%	1,87 %	13,8%	1,27 %	100,0%	0,00 %	100,0%	0,00 %
3	64,8%	2,43 %	12,1%	1,41 %	100,0%	0,00 %	100,0%	0,00 %
4	63,4%	3,48 %	10,9%	1,82 %	100,0%	0,00 %	100,0%	0,00 %
5	63,2%	3,51 %	10,8%	1,14 %	100,0%	0,00 %	100,0%	0,00 %
6	63,6%	3,88 %	10,4%	1,93 %	100,0%	0,00 %	100,0%	0,00 %
7	63,7%	4,72 %	9,3%	1,77 %	100,0%	0,00 %	100,0%	0,00 %
8	63,1%	4,26 %	9,2%	1,32 %	100,0%	0,00 %	100,0%	0,00 %
9	62,8%	4,60 %	8,6%	1,18 %	100,0%	0,00 %	100,0%	0,00 %
10	63,4%	3,20 %	8,6%	1,83 %	100,0%	0,00 %	100,0%	0,00 %
11	63,1%	4,18 %	9,1%	1,94 %	100,0%	0,00 %	100,0%	0,00 %
12	62,7%	3,29 %	8,7%	1,71 %	100,0%	0,00 %	100,0%	0,00 %
13	62,3%	3,68 %	9,1%	1,33 %	100,0%	0,00 %	100,0%	0,00 %
14	61,8%	3,26 %	9,7%	1,68 %	92,9%	0,31 %	77,2%	4,33 %
15	63,4%	4,63 %	10,6%	1,35 %	87,1%	0,65 %	35,1%	3,65 %
16	61,5%	1,12 %	10,2%	1,64 %	76,5%	0,81 %	28,7%	4,43 %
17	60,7%	4,75 %	10,1%	1,19 %	74,3%	1,01 %	24,6%	4,12 %
18	62,1%	3,36 %	9,0%	1,27 %	71,2%	1,70 %	19,3%	3,12 %
19	60,5%	3,73 %	9,2%	1,31 %	67,7%	1,31 %	18,2%	3,78 %
20	61,1%	3,50 %	9,7%	2,15 %	66,5%	1,40 %	14,1%	4,32 %
21	61,3%	4,86 %	9,6%	1,51 %	69,1%	1,56 %	16,3%	4,43 %
22	60,9%	3,73 %	9,2%	1,84 %	68,3%	1,27 %	14,8%	2,87 %
23	61,5%	4,58 %	8,7%	2,18 %	67,2%	0,98 %	14,4%	3,23 %
24	61,3%	3,17 %	9,2%	1,57 %	66,6%	1,34 %	12,1%	3,43 %
25	61,7%	4,92 %	8,6%	1,42 %	66,8%	1,48 %	11,7%	4,03 %
26	62,2%	3,98 %	8,7%	1,80 %	65,2%	1,60 %	11,2%	2,51 %
27	62,1%	3,04 %	8,2%	1,69 %	63,2%	1,77 %	11,6%	2,96 %
28	62,3%	3,01 %	7,9%	2,03 %	66,1%	0,89 %	10,1%	3,12 %
29	62,4%	2,86 %	8,3%	1,43 %	65,7%	1,54 %	10,6%	3,23 %
30	61,7%	3,06 %	7,8%	2,23 %	64,2%	2,02 %	10,3%	3,45 %
31	61,9%	2,98 %	7,7%	1,70 %	66,8%	0,57 %	10,9%	3,49 %
32	61,1%	3,48 %	7,5%	1,46 %	64,9%	0,95 %	10,5%	4,02 %
33	60,8%	3,16 %	7,9%	1,11 %	65,8%	1,62 %	11,3%	3,65 %
34	60,5%	3,23 %	7,3%	1,77 %	65,3%	1,34 %	10,8%	3,42 %
35	60,9%	3,87 %	7,6%	1,67 %	64,7%	1,29 %	10,1%	3,35 %
36	61,1%	2,94 %	8,0%	2,60 %	63,9%	1,65 %	9,8%	3,12 %
37	60,7%	3,53 %	7,7%	1,58 %	64,1%	1,33 %	9,2%	3,78 %
38	60,3%	3,28 %	7,2%	1,85 %	63,7%	1,42 %	10,0%	3,65 %
39	59,8%	3,28 %	7,1%	1,61 %	62,9%	1,32 %	9,7%	3,34 %
40	59,9%	3,78 %	7,3%	2,05 %	63,4%	0,97 %	9,3%	3,23 %

Tabela A.19: Nível de satisfação médio para colaboradores (c) e *free-riders* (f) para 10.000 nós, $c = 25\%$ e $c = 50\%$

Turno	$c = 75\%$	ϵ	$f = 25\%$	ϵ
1	100,0%	0,00 %	100,0%	0,00 %
2	100,0%	0,00 %	100,0%	0,00 %
3	100,0%	0,00 %	100,0%	0,00 %
4	100,0%	0,00 %	100,0%	0,00 %
5	100,0%	0,00 %	100,0%	0,00 %
6	100,0%	0,00 %	100,0%	0,00 %
7	100,0%	0,00 %	100,0%	0,00 %
8	100,0%	0,00 %	100,0%	0,00 %
9	100,0%	0,00 %	100,0%	0,00 %
10	100,0%	0,00 %	100,0%	0,00 %
11	100,0%	0,00 %	100,0%	0,00 %
12	100,0%	0,00 %	100,0%	0,00 %
13	100,0%	0,00 %	100,0%	0,00 %
14	100,0%	0,00 %	100,0%	0,00 %
15	100,0%	0,00 %	100,0%	0,00 %
16	100,0%	0,00 %	100,0%	0,00 %
17	100,0%	0,00 %	100,0%	0,00 %
18	100,0%	0,00 %	100,0%	0,00 %
19	96,2%	2,69 %	34,7%	0,33 %
20	87,4%	4,41 %	27,3%	1,21 %
21	79,1%	4,26 %	21,7%	1,45 %
22	78,7%	4,71 %	23,2%	1,13 %
23	77,3%	3,74 %	21,5%	0,92 %
24	76,4%	3,36 %	19,9%	1,38 %
25	75,7%	4,23 %	20,3%	1,69 %
26	72,1%	4,17 %	20,7%	1,72 %
27	71,7%	4,78 %	19,2%	1,43 %
28	71,2%	3,65 %	18,1%	1,38 %
29	71,6%	3,91 %	18,9%	1,11 %
30	71,2%	3,24 %	18,2%	1,62 %
31	71,5%	4,29 %	17,7%	1,43 %
32	71,1%	3,87 %	17,4%	1,84 %
33	70,8%	3,14 %	17,1%	1,79 %
34	70,6%	3,96 %	17,3%	1,97 %
35	71,2%	3,56 %	16,9%	2,12 %
36	69,9%	4,01 %	17,7%	2,01 %
37	69,3%	4,36 %	17,1%	1,24 %
38	69,2%	3,15 %	16,2%	1,72 %
39	68,6%	4,44 %	16,7%	1,69 %
40	68,4%	3,61 %	16,8%	2,41 %

Tabela A.20: Nível de satisfação médio para colaboradores (c) e *free-riders* (f) para 10.000 nós, $c = 75\%$

A.4 Estudo de Caso

	Cache		***Storage***		***Playpen***	
Execução	***Makespan* (s)**	ϵ	***Makespan* (s)**	ϵ	***Makespan* (s)**	ϵ
1	$5.277,43$	$2,21\%$	$6.772,18$	$1,24\%$	$6.821,14$	$3,62\%$
2	$3.775,60$	$3,47\%$	$3.877,43$	$4,21\%$	$6.838,85$	$2,98\%$
3	$3.882,58$	$2,67\%$	$3.974,81$	$4,36\%$	$6.784,93$	$3,01\%$
4	$3.899,74$	$3,08\%$	$3.892,20$	$3,89\%$	$6.795,93$	$1,92\%$
5	$3.892,20$	$4,33\%$	$3.941,10$	$4,11\%$	$6.809,67$	$2,43\%$

Tabela A.21: Comparação entre estratégias de armazenamento

	Colaborador		***Free-rider***	
Execução	***Makespan* (s)**	ϵ	***Makespan* (s)**	ϵ
1	$5.277,43$	$4,34\%$	$5426,639$	$4,87\%$
2	$3.775,60$	$4,18\%$	$3812,309$	$4,42\%$
3	$3.882,58$	$4,23\%$	$4006,185$	$4,63\%$
4	$4.399,74$	$4,01\%$	∞	-
5	$4.192,20$	$4,19\%$	∞	-

Tabela A.22: Medição do *makespan* entre colaborador e *free-rider*

	Colaborador		***Free-rider***	
Execução	***Nível de Satisfação***	ϵ	***Nível de Satisfação***	ϵ
1	$100,00\%$	$0,00\%$	$100,00\%$	$0,00\%$
2	$100,00\%$	$0,00\%$	$100,00\%$	$0,00\%$
3	$100,00\%$	$0,00\%$	$100,00\%$	$0,00\%$
4	$100,00\%$	$0,00\%$	$34,46\%$	$0,00\%$
5	$100,00\%$	$0,00\%$	$0,00\%$	$0,00\%$

Tabela A.23: Medição do nível de satisfação entre colaborador e *free-rider*