Escalonamento Tolerante a Sabotagem em Grades Computacionais Entre-Pares

Ana Cristina Alves de Oliveira[1], **Francisco Vilar Brasileiro**[1]

[1]Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Programa de Pós-Graduação em Ciência da Computação
Laboratório de Sistemas Distribuídos
Campina Grande - PB - Brasil

{cristina,fubica}@dsc.ufcg.edu.br

Abstract. *The computational grids were built to be an infrastructure to increase computing power. They have evolved in the sense of forming free-to-join communities to share computing power over the Internet, and won the title of peer-to-peer (P2P) grids. Note that as any user can freely join and leave a system, it can become susceptible to damages caused by cheater users. A solution to this problem is applying sabotage tolerant schemes, which are generally based on replication to estimate the computation correctness. This work concerns sabotage tolerant scheduling in P2P grids and shows that it is possible to obtain high levels of realiability for the computation results, minimizing the costs of replication through the use of credibility-based fault tolerance in the task scheduling.*

Resumo. *As grades computacionais foram construídas para servirem de infra-estrutura para o aumento da potência computacional. Elas evoluíram no sentido de formarem comunidades de livre ingresso para compartilhamento de potência computacional sobre a Internet e ganharam o título de grades entre-pares (P2P). Note que o fato de qualquer usuário poder ingressar e sair livremente de um sistema, pode torná-lo suscetível a danos causados por usuários trapaceiros. Uma solução para este problema é aplicar técnicas de tolerância a sabotagem, que geralmente são baseadas em replicação para estimar a corretude da computação. Este trabalho aborda escalonamento tolerante a sabotagem em grades P2P e demonstra que é possível obter altos níveis de confiabilidade para os resultados da computação, minimizando custos de replicação por meio do emprego de tolerância a faltas baseada em credibilidade no escalonamento de tarefas.*

1 Introdução

As grades computacionais, também conhecidas na literatura por "*grids*", foram desenvolvidas para proporcionar uma infra-estrutura para execução de aplicações paralelas em máquinas geograficamente dispersas, de modo que um usuário que se conecte à grade tenha acesso a potência computacional como se estivesse ligando-se a uma tomada para receber energia elétrica. No intuito de compartilhar ciclos computacionais, surgiram as grades entre-pares (ou *peer-to-peer* - P2P). Nestas, os usuários são livres para ingressarem, doarem ciclos ociosos de CPU e executarem suas próprias aplicações paralelas. A

comunidade OurGrid [Cirne et al. 2006] é um exemplo de grade P2P e a Figura 1 mostra um exemplo de submissão de trabalhos (*jobs*) do usuário para máquinas dessa grade. O usuário utiliza um escalonador de aplicação (*broker*) que submete seus trabalhos a um *peer* responsável por lhe prover máquinas de forma transparente (de seu próprio sítio ou recebidas da comunidade). De posse das máquinas, o *broker* faz o escalonamento das tarefas, além disso informa ao usuário sobre o estado das mesmas, erros na execução e quando o trabalho está concluído ou falho.

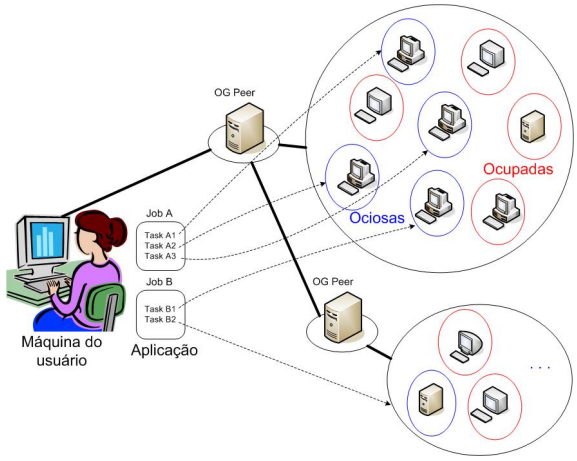

Figure 1. Alocação das tarefas do usuário nas máquinas da grade.

As grades P2P estão suscetíveis a um problema já diagnosticado em sistemas de computação voluntária, como o SETI@home [Molnar 2000]: a possibilidade de usuários trapaceiros (por se ter de confiar em máquinas desconhecidas). Técnicas para evitar sabotadores foram propostas na literatura, de modo a tentar contornar ameaças concernentes à execução de processos em máquinas não confiáveis. Alguns mecanismos de proteção de código para combater engenharia reversa e alteração maliciosa de código são citados a seguir:

- **Ofuscação:** é um mecanismo que insere trechos de código confusos, mantendo as mesmas funcionalidades do software para evitar que o código seja entendido [Collberg and Thomborson 2002] [Collberg et al. 1997] [Sarmenta 1999];
- **Computação criptografada:** propõe a aplicação de técnicas de criptografia ao código ou aos dados, no intuito de evitar a aprendizagem de funcionalidades e manter a confidencialidade das informações [Dinda 2004] [Hohl 1998] [Sarmenta 1999];
- **Adição de marcas d'água ou códigos de certificação:** esta técnica introduz trechos que possibilitam a detecção de alterações no código. Ela pode ser útil para fazer o programa se comportar erroneamente em caso de sabotagem [Collberg and Thomborson 2002];

Embora interessantes, essas técnicas podem não ser passíveis de implementação, como no caso de ofuscação [Barak et al. 2001], ou apresentarem custos muito altos que inviabilizem seu emprego em grades computacionais, como algoritmos de criptografia que tenham um custo computacional maior do que o da execução de algumas tarefas submetidas à grade. Além do mais, perdem em generalidade, uma vez que devem ser implementadas pelo desenvolvedor da aplicação na maioria dos casos.

Na tentativa de reduzir custos e aumentar o escopo de aplicações, este artigo apresenta um esquema de escalonamento de aplicações *bag-of-tasks* (BoT - aplicações paralelas cujas tarefas são independentes entre si) que limita a probabilidade de erro na aceitação de tarefas e minimiza o custo computacional, em termos da redundância empregada, e o *makespan* (tempo necessário para executar um *job*). Neste trabalho, procura-se obter tolerância a sabotagem utilizando mecanismos de alto nível baseados em reputação para solucionar probabilisticamente o problema de usuários maliciosos. Para tal, a técnica de *tolerância a faltas baseada em credibilibidade* proposta em [Sarmenta 2002] para tolerar sabotagem em sistemas de computação voluntária foi inserida no mecanismo de escalonamento de grades P2P.

O processo de estimativa das reputações é explicado na Seção 2. A Seção 3 apresenta como se pode inserir as técnicas propostas por Sarmenta no mecanismo de escalonamento de tarefas em grades P2P. A avaliação de desempenho do serviço de escalonamento é feita na Seção 4 e na 5 são discutidos trabalhos relacionados. Por fim, as considerações finais e trabalhos futuros encontram-se na Seção 6.

2 Tolerância a Sabotagem

Em [Sarmenta 2002], Sarmenta propõe mecanismos para diminuição de custos de tolerância a sabotagem a partir da combinação de **votação** e ***spot-checking*** (uma técnica que testa as máquinas com tarefas de averiguação, ou *spot-checks*, cujos resultados já são conhecidos ou existe alguma maneira confiável para os calcular). Baseia-se na premissa de que o resultado de uma tarefa *somente* pode ser aceito como correto após sua *credibilidade* atingir, no mínimo, uma probabilidade condicional média θ de seu resultado estar correto, denominada *limiar de credibilidade* ou *credibilidade-alvo*. Esta prática permite que altas taxas de acerto sejam obtidas com baixos níveis de redundância, se comparados à votação majoritária tradicional.

Neste trabalho, a tolerância a sabotagem utiliza equações propostas em [Sarmenta 2002] para calcular a credibilidade de quatro objetos relevantes do sistema. Os objetos levados em consideração para fins do cálculo da credibilidade final do trabalho na grade são: 1) máquinas (Cr_P); 2) resultados (Cr_R); 3) grupos de resultados (Cr_G) e 4) tarefas (Cr_W). Os métodos adotados para calcular as credibilidades estão descritos nas próximas seções.

2.1 Modelo do Sistema

Assume-se neste trabalho que todas as tarefas (*tasks*) são independentes (BoT), calculadas deterministicamente e submetidas em um grupo chamado *job*, que constitui uma aplicação paralela. O que determina se uma tarefa está concluída, ou não, é sua credibilidade que deve atingir o *limiar de credibilidade* θ desejado pelo usuário do sistema. Para especificar o limiar, o usuário define uma propriedade constante chamada *taxa de erro aceitável* ε_{acc}, que corresponde à máxima taxa de erro aceitável pelo sistema. O limiar de credibilidade, $\theta = 1 - \varepsilon_{acc}$, é o princípio básico da *tolerância a faltas baseada em credibilidade* [Sarmenta 2002].

Outro parâmetro importante do sistema é a *fração de faltosos* f, também chamada de fração de sabotadores ou de máquinas maliciosas. Cada sabotador é modelado como um processo de Bernoulli com uma *taxa de sabotagem* s, a qual exprime a probabilidade

da máquina retornar um resultado incorreto. Ainda na configuração do sistema, segundo Sarmenta, deve ser estabelecida a *taxa de geração de spot-checks* q que é usada para informar a probabilidade com que uma tarefa deste tipo deve ser escalonada.

Assume-se que a grade possui ao menos 1 sítio (*peer*) confiável. Isto significa que suas máquinas não passam pelo processo de *spot-checking*, nem uma tarefa que execute em uma máquina confiável necessita ser replicada, pois a credibilidade de seu resultado é tida como 100%. Todas as máquinas não oriundas de sítios confiáveis, continuam sendo avaliadas por *spot-checks* e tendo seus resultados submetidos à votação, independentemente de seus valores de credibilidade.

2.2 Credibilidade das Máquinas

Para calcular a credibilidade de uma máquina, usa-se o número de spot-checks k em que ela foi aprovada. Ou seja, pode-se estimar como uma máquina tende a retornar um resultado correto pelo número de *spot-checks* pelos quais passou. Quanto mais *spot-checks* ela passar, mais confiança se tem de que a máquina é correta ou, no mínimo, que ela não tem uma taxa de sabotagem s alta. Também não é preciso considerar as credibilidades das máquinas que foram identificadas retornando um resultado incorreto, porque elas são consideradas sabotadores e banidas do sistema.

A Equação 1 representa a fórmula do cálculo da credibilidade para uma máquina P utilizando as técnicas de ***s**pot-**c**hecking* e **l**ista **n**egra (*scln*), onde f representa a fração de máquinas maliciosas (ou faltosas). Esta credibilidade é um limite inferior estrito para a probabilidade de uma máquina P retornar um resultado correto.

$$Cr_P(P)_{scln} = 1 - \frac{sf(1-s)^k}{(1-f) + f(1-s)^k} \tag{1}$$

2.3 Credibilidade dos Resultados

O método seguido para estimar a credibilidade de um resultado R é simples e está expresso na Equação 2. Apenas se assume que é igual à credibilidade da máquina *R.solucionador* que o produziu.

$$Cr_R(R) = Cr_P(R.solucionador) \tag{2}$$

É possível distinguir a credibilidade do resultado da credibilidade do solucionador. Por exemplo, para evitar sabotadores que dão resultados corretos no início com a intenção de ganhar credibilidade e, quando sabem que sua credibilidade está alta, começam a retornar mais resultados incorretos, pode-se usar uma política de atribuir uma credibilidade menor do que a da máquina aos últimos resultados recebidos. Por exemplo, para calcular a credibilidade de um resultado, poderia ser usada uma janela deslizante que conteria os últimos x resultados recebidos e a credibilidade seria calculada por uma média ponderada com maior peso para a credibilidade dos resultados mais recentes.

2.4 Credibilidade dos Grupos de Resultados e das Tarefas

Com o mecanismo de replicação, vários resultados de uma mesma tarefa poderão ser obtidos. Para estimar a credibilidade de uma tarefa, primeiro se agrupam os resultados iguais e depois se estima a credibilidade de cada grupo de resultados.

Se uma tarefa W **tem apenas um resultado** R, então se pode estimar a credibilidade da tarefa facilmente, como mostra a Equação 3.

$$Cr_W(W) = Cr_R(R) = Cr_P(R.solucionador) \tag{3}$$

Como mais resultados podem ser produzidos, dividem-se os resultados em g grupos e um grupo é representado por G_a, onde $1 \leq a \leq g$. Cada grupo G_a possui m_a membros (resultados). As credibilidades dos grupos de resultados se baseiam na probabilidade condicional de correção, dada a combinação dos resultados recebidos até o momento do cálculo, como mostrado na Equação 4.

$$\begin{aligned} Cr_G(G_a) &= \frac{P(G_a \text{ correto})P(\text{todos os outros são incorretos})}{P(\text{pegue } g \text{ grupos, onde cada } G_a \text{ tem } m_a \text{ membros})} \\ &= \frac{P(G_a \text{ correto}) \prod_{i \neq a} P(G_i \text{ incorreto})}{\prod_{j=1}^{g} P(G_j \text{ incorreto}) + \sum_{j=1}^{g} P(G_j \text{ correto}) \prod_{i \neq j} P(G_i \text{ incorreto})} \end{aligned} \tag{4}$$

Dado que a:

- probabilidade de **todos** os resultados R_{ai} do grupo G_a serem corretos é: $P(G_a \text{ correto}) = \prod_{i=1}^{m_a} Cr_R(R_{ai})$
- probabilidade de **todos** os resultados R_{ai} do grupo G_a serem incorretos é: $P(G_a \text{ incorreto}) = \prod_{i=1}^{m_a}(1 - Cr_R(R_{ai}))$

Se um grupo de resultados alcançar o limiar de credibilidade, então este resultado será aceito e poderá ser considerado um *spot-check* passado por todas as máquinas que o produziram. Em contrapartida, as máquinas dos demais grupos serão tratadas como detectadas retornando um resultado errado em um *spot-check*. A credibilidade da tarefa será a credibilidade do grupo vencedor.

2.5 Aplicação das Credibilidades

Uma vez que se têm métodos para calcular a credibilidade dos objetos do sistema, precisa-se definir como aplicar tais métodos. Sarmenta apresenta duas maneiras interessantes de utilizar as credibilidades: votação e *spot-checking* combinados e *spot-checking* por votação.

A união de *spot-checking* e votação funciona aproveitando *spot-checking* e votação em um único esquema e aproveita a baixa redundância do *spot-check* (com suas taxas de erro que decrescem linearmente no tempo) e a precisão da votação (que diminui exponencialmente as taxas de erro à medida que o número de réplicas aumenta). Neste mecanismo, caso a fila de tarefas seja grande e o limiar de credibilidade θ seja baixo, as máquinas aumentarão suas credibilidades sendo aprovadas em *spot-checks* e seus resultados poderão ser aceitos sem a necessidade de votação. Assim, o grau de replicação poderá ser menor do que 2. Contudo, se essa condição não ocorrer, inicia-se a alocação de tarefas redundantes às máquinas e, com isto, o mecanismo de votação dos resultados.

A técnica de *spot-checking* **por** votação segue o mesmo princípio da combinação de *spot-checking* **e** votação. A diferença entre elas é que quando um resultado é aceito na votação baseada em credibilidade, o seu resultado é considerado um *spot-check*. Dessa feita, as máquinas que contribuíram para o resultado têm seu número de *spot-checks* aprovados k acrescido de 1 e, em decorrência, sua credibilidade aumentada. A propósito, as máquinas que perderam a votação são consideradas sabotadores. A redundância será no mínimo 2, devido à votação. É interessante perceber que, diferentemente da votação majoritária cujo *quorum* é no mínimo 3, esta votação pode ser feita com apenas 2 resultados, pois a credibilidade é o critério de eleição. Esta técnica implica um aumento mais freqüente da credibilidade das máquinas corretas, tornando a votação mais rápida, e uma conseqüente diminuição da taxa de erro. Vale ressaltar que nos casos em que a redundância deve obrigatoriamente ser menor do que 2, esta técnica não apresenta vantagens. Além disso, para utilizar votação baseada em credibilidade, deve-se esperar que as máquinas ganhem certa credibilidade antes de iniciar a votação, para evitar que usuários maliciosos conluiam e eliminem os corretos na votação.

3 Escalonamento Considerando Sabotagem

Há uma parceria entre o sistema de reputação e o algoritmo de escalonamento. O primeiro calcula e armazena as credibilidades das máquinas e das tarefas que serão levadas em consideração no escalonamento pelo segundo. Em contrapartida, as tarefas escalonadas são utilizadas para alimentar o sistema de reputação e fornecer os subsídios requeridos para que este possa calcular as credibilidades.

3.1 Algoritmo de Escalonamento

O trabalho de Sarmenta adota um modelo em que todas as tarefas são oriundas de um único servidor (mestre) que as atribui seguindo uma fila circular em rodízio para garantir uma distribuição eqüitativa de trabalho entre as máquinas. Este modelo foi adaptado neste trabalho para operar no contexto de escalonamento em grades P2P. Introduziram-se as técnicas para estimação de reputação propostas por Sarmenta no serviço de escalonamento de grades P2P, de modo a apenas serem aceitas tarefas cujas credibilidades são superiores ao limiar de credibilidade definido, como mostra o Algoritmo 1. Ele termina quando todas as tarefas atingem a credibilidade-alvo.

Algoritmo 1 Escalonamento adaptando o modelo de Sarmenta

se máquina M for local **então**
  $Cr_P(M) = 1$
senão
  $Cr_P(M) = 0$
fim se
enquanto (houver máquina ociosa) $\wedge$ (houver tarefa que não atingiu a Cr_{alvo}) **faça**
  escolha uma máquina aleatoriamente
  pegue a tarefa corrente
  escalone
  selecione a próxima tarefa fila
fim enquanto

O mecanismo de *spot-checking* funciona à parte. Note que as máquinas locais são consideradas confiáveis e seus resultados são adotados como corretos e armazenados

como *spot-checks*. Enquanto alguma tarefa não for executada por uma máquina confiável, não haverá *spot-checks* no sistema. A taxa de geração de *spot-checks* q define a probabilidade com que uma tarefa deste tipo deve ser escalonada. Por exemplo, se $q = 0,10$, então a probabilidade de uma máquina receber um *spot-check* no escalonamento é de 10%. Quando uma tarefa executa em uma máquina local ou em uma máquina que atingiu a credibilidade-alvo (portanto, considerada confiável), a tarefa e o resultado obtido são armazenados para posterior escalonamento.

3.2 Sistema de Reputação

O sistema de reputação é atualizado com base nos resultados das execuções das tarefas. Para avaliar a correção das tarefas, os mecanismos de verificação de resultados escolhidos são votação baseada em credibilidade e *spot-checking* **por** votação, conforme descrito a seguir: (a) inicialmente, todos os nós locais têm credibilidade 1 e os não-locais têm credibilidade 0; (b) os nós, com exceção dos nós locais, ganham credibilidade ao serem aprovados em tarefas de averiguação (*spot-checks*); (c) quando a credibilidade é superior a um certo limiar e se têm, no mínimo, dois resultados de alguma tarefa, pode-se votar nos resultados. Aumenta-se a credibilidade das máquinas que produziram os resultados ganhadores da votação e se remove do sistema (adicionando-se em lista negra) os nós que perderam; (d) armazenam-se as credibilidades obtidas e no momento em que uma tarefa atingir o limiar esperado, será considerada concluída e não mais será replicada.

Existem ataques que anulam a utilidade da lista negra, como:

- **ataque de *whitewashing*:** os nós deixam o sistema e ingressam com nova identidade.
- **ataques de *Sybil*:** um mesmo trapaceiro ingressa na grade com mais de uma identidade para burlar o sistema de reputação, uma vez que não há um número definido de vezes que uma máquina pode votar.

Sarmenta [Sarmenta 2002] também propõe uma equação particular para estimar a credibilidade de uma máquina na ausência de lista negra, inclusive apresentando bons resultados em termos de redundância e erro. Entretanto, não estão no escopo deste artigo. Uma forma de minimizar tais ataques seria aumentar o custo de um usuário ingressar no sistema, como, por exemplo, impor períodos de permanência mínimos, utilizar informação que identifique a máquina do usuário ou empregar um esquema de autenticação confiável.

3.3 Componentes Principais

Cinco componentes principais constituem o núcleo do serviço de escalonamento tolerante a sabotagem proposto: 1) verificador de resultados; 2) calculador de credibilidade; 3) módulo para armazenamento de credibilidade; 4) provedor de *spot-checks* e 5) lista negra. A Figura 2 mostra as interações entre eles.

No momento em que o escalonador é notificado de que uma máquina completou a réplica de uma tarefa, o módulo *calculador de credibilidade* estima qual a credibilidade daquela tarefa e tal credibilidade, juntamente com a da máquina é armazenada no componente de *armazenamento de credibilidade*. O cálculo da credibilidade de uma tarefa é feito com base na credibilidade dos resultados de suas réplicas recebidos até aquele momento, conforme descrito na Seção 2.4. Os resultados iguais são postos em um mesmo

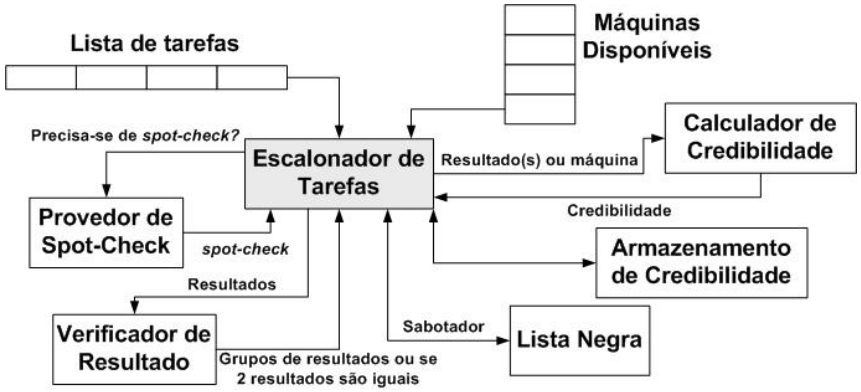

Figure 2. Interações entre os componentes do serviço de escalonamento.

grupo e o grupo que atingir a credibilidade-alvo ganha a votação. Enquanto a credibilidade da tarefa não atingir o limiar esperado, a tarefa continuará no estado de incompleta e será sujeita a novas replicações.

Se um sabotador for identificado devolvendo um resultado incorreto pelo componente *verificador de resultados*, ele será removido do sistema e todos os resultados que forneceu serão descartados. Para evitar o recebimento de resultados oriundos de sabotadores, existe uma *lista negra* onde um identificador da máquina traidora é armazenado.

4 Resultados Obtidos

4.1 Metodologia

Os resultados foram obtidos por meio de simulações do algoritmo de escalonamento apresentado na Seção 3. A grade é composta por 1.000 máquinas distribuídas igualmente entre 100 *peers* e cada simulação consiste no escalonamento de 1 *job* com 6.000 tarefas.

Executaram-se simulações para um ambiente de grade sem a presença de usuários trapaceiros, em que todas as máquinas são confiáveis, para se obter o tempo **ideal** necessário para completar a execução de um *job*. Neste caso, a redundância é 1, pois o trabalho é executado apenas 1 vez e a probabilidade de erro é 0% (credibilidade dos resultados = 100%).

Um conjunto de cenários, variando a taxa de sabotagem, o percentual de máquinas confiáveis e a taxa de erro aceitável foram simulados. Para todos estes cenários, foram usados uma taxa de geração de *spot-checks* constante ($q = 10\%$) e uma fração inicial de máquinas trapaceiras padrão ($f = 30\%$). A redundância (a razão entre número de tarefas executadas e o número original de tarefas) e o *makespan* foram as métricas observadas nesta avaliação e devidamente comparadas com o caso ideal.

4.2 Resultados Preliminares

A Figura 3 demonstra que as técnicas de tolerância a faltas baseadas em credibilidade aplicadas no escalonamento de tarefas permitem atingir, por exemplo, a taxa de erro aceitável ε_{acc} de $0,00001 = 0,001\%$ com menos de 4 réplicas para cada tarefa em um ambiente em que apenas 1% das máquinas são confiáveis e sabotadores retornam resultados incorretos em 25% dos casos. É importante notar que quanto maior é a taxa de sabotagem,

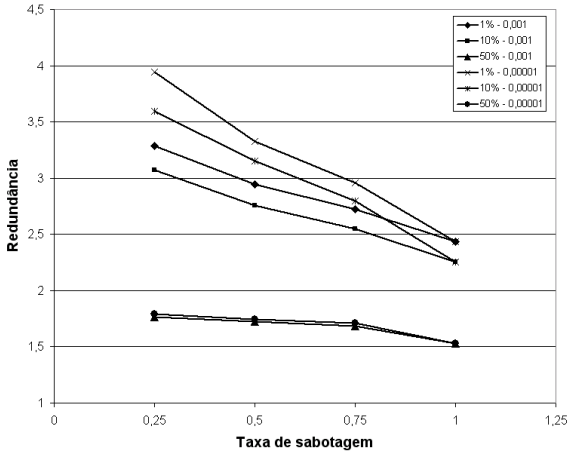

Figure 3. Redundância média vs. taxa de sabotagem com $1\%, 10\%, 50\%$ de máquinas confiáveis e $\varepsilon_{acc} = 0,001; 0,00001$.

menor será o grau de replicação, pois maior é a chance do trapaceiro ser pego em um *spot-check* ou na votação e ser adicionado a lista negra. Por exemplo, para se obter um $\varepsilon_{acc} = 0,00001$ com uma taxa de sabotagem s de 100%, é necessária apenas uma redundância média de cerca de $2,4$.

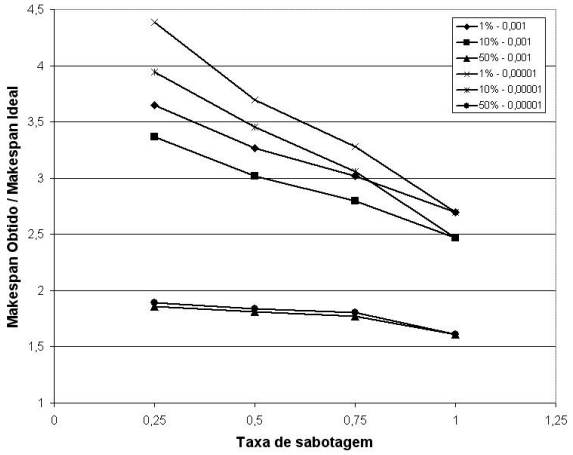

Figure 4. Razão entre *makespan* médio obtido e *makespan* médio ideal vs. taxa de sabotagem com $1\%, 10\%, 50\%$ de máquinas confiáveis e $\varepsilon_{acc} = 0,001; 0,00001$.

Quanto ao tempo para completar a execução do *job* (*makespan*), pode-se observar pela Figura 4 que o tempo médio obtido em relação ao tempo médio ideal é proporcional à redundância. Apesar de serem próximos, não são totalmente equivalentes por causa da sobrecarga gerada pelos mecanismos do serviço de escalonamento. Tanto para a redundância, quanto para o *makespan*, quando se aumenta o número de máquinas confiáveis, percebe-se que as tarefas tendem a alcançar a credibilidade-alvo mais rapidamente e, portanto, requerem pouco trabalho redundante para garantirem o erro aceitável ε_{acc}.

Em geral, ao término da execução do *job* foram obtidas taxas de erro bem menores do que o erro aceitável ε_{acc} definido, como mostram as Figuras 5 e 6. O gráfico da Figura 5

representa a taxa de erro obtida, empregando-se a redundância necessária para garantir um $\varepsilon_{acc} = 0,001$ (= 1×10^{-03}), com 50%, 10% e 1% de máquinas confiáveis, respectivamente. Enquanto que a Figura 6 garante um erro aceitável 100 vezes menor ($\varepsilon_{acc} = 0,00001 = 1 \times 10^{-05}$). Perceba que: 1) as taxas de erro obtidas e a redundância convergem à medida que o número de máquinas confiáveis aumenta. Por exemplo, quando a taxa de máquinas confiáveis é de 50%, independentemente de s, todos os erros são inferiores a 1×10^{-05} para $\varepsilon_{acc} = 1 \times 10^{-03}$ e a 1×10^{-07} para $\varepsilon_{acc} = 1 \times 10^{-05}$ e as redundâncias próximas a $1,75$; 2) as menores taxas de erro (próximas a 0%) e de redundância (ver Figura 3) são obtidas quando $s = 100\%$; 3) em todos os casos, o erro obtido é menor do que ε_{acc}.

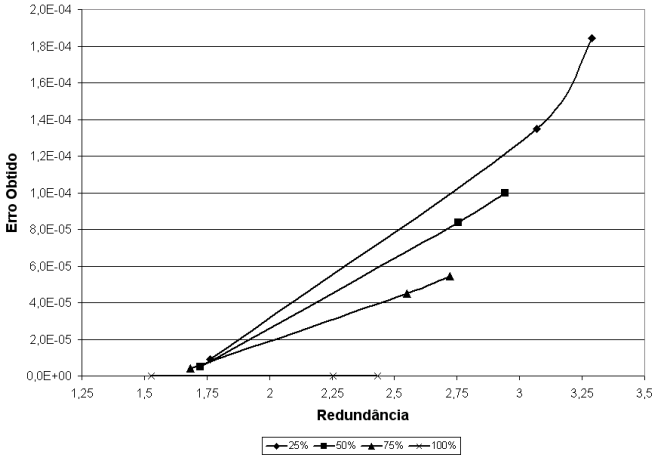

Figure 5. Redundância vs. erro obtido para s =25%, 50%, 75%, 100%; $\varepsilon_{acc} = 0,001$ com 50%, 10% e 1% de *peers* confiáveis.

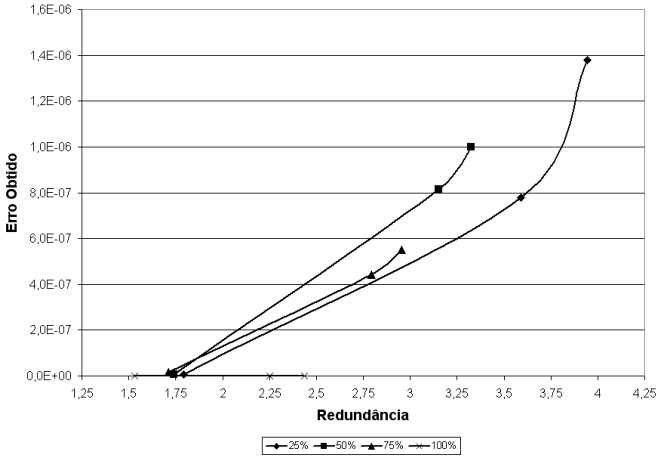

Figure 6. Redundância vs. erro obtido para s =25%, 50%, 75%, 100%; $\varepsilon_{acc} = 0,00001$ com 50%, 10% e 1% de *peers* confiáveis.

5 Trabalhos Relacionados

O trabalho de Sarmenta [Sarmenta 2002] é o fundamento teórico das técnicas de verificação de resultados adotadas neste trabalho. Todavia, escalonamento está fora de seu escopo. Uma das contribuições desta pesquisa é adaptar seu trabalho de modo a integrá-lo em um serviço de escalonamento aplicado a grades computacionais P2P, haja vista

que ele foi originalmente concebido para sistemas de computação voluntária. Sarmenta propõe basicamente três técnicas de tolerância a sabotagem: **votação**, ***spot-checking*** e **tolerância a faltas baseada em credibilidade**. A última é genérica e permite combinar vários mecanismos de verificação de resultados. Sarmenta apresenta bons resultados para a combinação de votação e de *spot-checking*.

Um importante trabalho relacionado é o de Zhao et al. [Zhao et al. 2005]. Ele propõe o uso de **replicação** (votação majoritária) e de esquemas de ***quizes*** para verificação da correção dos resultados. A técnica de *quizes* é bastante semelhante a *spot-checking*, diferindo apenas que enquanto *spot-checks* são tratados como tarefas enviadas independentemente com uma determinada probabilidade, os *quizes* são enviados em um pacote com outras tarefas normais, devendo ser executadas todas juntas. As tarefas do pacote só são aceitas como corretas se todos os *quizes* estiverem corretos. Este projeto foi pioneiro a unir técnicas de estabelecimento de reputações à estratégia de escalonamento de grades computacionais P2P, propondo um arcabouço para **escalonamento baseado em confiança**.

Uma característica importante do trabalho de Zhao et al. é que a razão de *quiz* e o fator de replicação são adaptativos. Existe um *módulo verificador de resultados* que executa uma **função de razão de *quiz*** e uma **função de fator de replicação** dado o valor da reputação de um trabalhador (máquina). Quanto mais confiável for a máquina, menores serão a razão de *quiz* e o fator de replicação. Porém, não entra em detalhes de como essas funções devem ser calculadas e deixa a cargo do desenvolvedor a tarefa de escolher um sistema de reputação para estimar as credibilidades.

Outro trabalho encontrado é o de Sonnek et al. [Sonnek et al. 2006]. Ele propõe apenas o uso de *votação majoritária*, mas com algoritmos de escalonamento que procuram satisfazer uma *probabilidade de correção alvo* (LOC – *likelihood of correctness*). Propuseram quatro algoritmos para formar grupos de redundância que executam uma mesma tarefa (*First-Fit*, *Best-Fit*, *Random* e *Fixed*), procurando satisfazer o LOC alvo. No intuito de limitar as taxas de replicação, adotam um tamanho mínimo ou máximo para os grupos. O resultado será aceito quando for majoritário no grupo de redundância. Entretanto, não mencionam uma maneira eficiente de estimar os tamanhos máximos e mínimos dos grupos de redundância.

Sabe-se que a técnica de votação majoritária apresenta altas taxas de replicação (no mínimo, o trabalho deve ser feito três vezes). Os esquemas de *quizes*, por sua vez, são escalonados em forma de um pacote com várias tarefas onde se percebe que se algum *quiz* estiver incorreto haverá um desperdício de computação útil para todas as tarefas do pacote também. Apesar da redundância aumentar a probabilidade de serem aceitos resultados corretos, ela significa desperdício de recursos.

6 Considerações Finais e Trabalhos Futuros

Os trabalhos relacionados à área, em geral, ou não tratam de escalonamento ou empregam mecanismos baseados em votação majoritária que apresentam altos índices de replicação. O diferencial deste trabalho é a proposta de um mecanismo de escalonamento tolerante a sabotagem para grades P2P que está aliado a um sistema de reputação com técnicas que requerem baixa redundância e possibilitam atingir baixas taxas de erro na aceitação de tarefas.

Os resultados obtidos comprovam a idéia de que a técnica de tolerância a faltas baseada em credibilidade proposta por Sarmenta pode ser empregada para obter tolerância a sabotagem em ambientes de grades computacionais P2P. Todavia, alguns tipos de ataques não foram tratados, os quais se pretende abordar em trabalhos futuros. Espera-se ainda considerar a fração de sabotadores f e a taxa de sabotagem s como variáveis desconhecidas e propor métodos para estimá-las, bem como utilizar uma taxa de geração de *spot-checks* dinâmica que se adapte à credibilidade da máquina. Espera-se ainda aprimorar o algoritmo de escalonamento estudado, a partir da a simulação de novas heurísticas de escalonamento. Além disso, planeja-se implementar este serviço em um ambiente de grades P2P em produção e o avaliar.

References

Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., and Yang, K. (2001). On the (im)possibility of obfuscating programs. *CRYPTO*.

Cirne, W., Brasileiro, F., Andrade, N., Costa, L., Andrade, A., Novaes, R., and Mowbray, M. (2006). Labs of the world, unite!!! *Journal of Grid Computing*.

Collberg, C. and Thomborson, C. (2002). Watermarking, tamper-proofing, and obfuscation tools for software protection. *IEEE Transactions on Software Engineering*, 28(8).

Collberg, C., Thomborson, C., and Low, D. (1997). A taxonomy of obfuscating transformations. Technical report 161, Department of Computer Science, The University of Auckland, New Zealand.

Dinda, P. (2004). Addressing the trust asymmetry problem in grid computing with encrypted computation. In *Proceedings of the Seventh Workshop on Languages, Compilers and Run-time Support for Scalable Systems (LCR)*.

Hohl, F. (1998). *Mobile Agents and Security*, chapter Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts. Springer Verlag, Germany.

Molnar, D. (2000). The seti@home problem.

Sarmenta, L. F. G. (1999). Protecting programs from hostile environments: Encrypted computation, obfuscation, and other techniques. *Area Exam Paper, Dept. of Electrical Engineering and Computer Science, MIT*.

Sarmenta, L. F. G. (2002). Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems (Expanded journal version of CCGrid '01 Best Paper Finalist paper), Elsevier*.

Sonnek, J., Nathan, M., and Chandra, A. (2006). Reputation-based scheduling on unreliable distributed infrastructures. To appear in the Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS'06), Lisboa, Portugal, July 2006.

Zhao, S., Lo, V., and GauthierDickey, C. (2005). Result verification and trust-based scheduling in peer-to-peer grids. In *Fith IEEE International Conference on Peer-to-Peer Computing (IEEE P2P)*, pages 31–38.