Um Modelo de Diagnóstico Distribuído e Hierárquico para Tolerância a Ataques de Manipulação de Resultados em Grades Computacionais

**Felipe Martins, Rossana M. de Castro Andrade,
Aldri L. dos Santos, José Neuman de Sousa**

GREat – Grupo de Redes de Computadores, Engenharia de Software e Sistemas
Universidade Federal do Ceará (UFC)
Bloco 910 – Campus do Pici – 60.455-760 – Fortaleza – CE – Brazil

felipe@cenapadne.br,{rossana,aldri,neuman}@lia.ufc.br

Abstract. *Grid applications are susceptible to manipulation attacks, since nodes can act in a malicious way, corrupting the jobs results. In order to avoid that grid users obtain uncorrected results, this paper presents a diagnosis model for verifying integrity of the jobs processing in computational grids. The model establishes a hierarchy among the nodes, in accordance to the historical behavior of the nodes in the environment. A new security layer is implemented in a grid simulator in order to validate this strategy. The results testify the effectiveness of the model using scenarios with different quotas of malicious nodes, providing a rate of 100% detection and 99,7% accuracy of processed jobs.*

Resumo. *As aplicações em grades estão suscetíveis a ataques de manipulação, uma vez que os nós podem agir de forma maliciosa, corrompendo os resultados dos jobs processados. Para evitar que usuários obtenham resultados incorretos, este trabalho propõe um modelo de diagnóstico para verificação de integridade na execução dos jobs em grades computacionais. O modelo estabelece uma hierarquia de acordo com o histórico comportamental dos nós no ambiente. Para validar a estratégia, foi implementada uma nova camada de segurança em um simulador de grades. Os resultados obtidos atestam a eficácia do modelo em cenários com diferentes taxas de nós maliciosos, oferecendo um índice de detecção de 100% e acurácia de 99,7% dos jobs processados.*

1. Introdução

A segurança da informação em grades computacionais envolve requisitos que vão além dos estabelecidos para as redes convencionais. Tratando-se especificamente de *integridade*, a maioria das soluções existentes resolve essa questão apenas no escopo de transmissão, garantindo a não-violação dos dados durante a comunicação entre as máquinas [1]. Todavia, é preciso garantir também a integridade dos dados durante o processamento, de modo que os resultados das tarefas (jobs) processadas em uma grade não sofram qualquer alteração. Caso contrário, a manipulação de resultados compromete a aplicação como um todo, incidindo num alto custo em termos de desempenho. Por outro lado, algoritmos de diagnóstico em nível de sistema são comumente utilizados como uma estratégia de tolerância a falhas e podem ser aplicados em diferentes tipos de redes, a fim de se conhecer quais unidades estão falhas e quais estão em pleno funcionamento [2].

Esta abordagem tanto pode ser realizada de forma autônoma, como também considera a natureza heterogênea e dinâmica do ambiente, características intrísecas às grades.

Para evitar que usuários obtenham resultados incorretos em virtude de elementos maliciosos, este artigo apresenta um modelo de diagnóstico para tolerância a falhas de segurança em grades computacionais, abordando a verificação de integridade dos jobs. Desta forma, é possível excluir as unidades de processamento (nós) de má conduta, oferecendo, portanto, um ambiente de computação de alto desempenho formado apenas por nós confiáveis. Este artigo está organizado da seguinte maneira: a seção 2 apresenta uma taxonomia dos nós que apresentam mau comportamento e uma visão geral sobre diagnóstico em nível de sistema; a seção 3 descreve o modelo de diagnóstico para tolerância a ataques de manipulação em grades; os cenários de simulação e os resultados obtidos são discutidos na seção 4; e a seção 5 traz as considerações finais e os trabalhos futuros.

2. Nós com Mau Comportamento e Diagnóstico em Nível de Sistema

As falhas ocasionadas por mau comportamento são classificadas na literatura como bizantinas. No entanto, as falhas de mau comportamento podem ser categorizadas em três classes, uma vez que envolvem componentes (nós) que passam a agir de maneira *inativa*, *egoísta* ou *maliciosa* [3]. Os nós inativos não cooperam com a rede, deixando de encaminhar pacotes ou omitindo informações sobre seus recursos. Os nós egoístas negligenciam ajuda aos demais nós, favorecendo apenas seus próprios interesses. Quanto aos nós maliciosos, estes possuem, por exemplo, interesse em subverter os recursos da grade, oferecer um resultado inválido, ou mesmo difundir vírus entre as máquinas do ambiente.

A classe de nós maliciosos pode ainda ser subdividida em três tipos: *tolos*, *comuns* e *inteligentes*. Nós tolos sempre retornam resultados arbitrários; nós comuns retornam resultados arbitrários com uma certa probabilidade; nós inteligentes agem normalmente durante um período, até que passam deliberadamente a retornar resultados arbitrários com uma certa probabilidade. Dentre as classificações de mau comportamento discutidas, as falhas de natureza maliciosa que geram resultados corrompidos, em especial os nós maliciosos tolos e comuns, constituem o escopo deste trabalho.

Algoritmos de diagnóstico em nível de sistema definem uma série de testes, cujo conjunto de respostas (*síndrome*) permite identificar as unidades que estão falhas. Dentre os principais modelos de diagnóstico, destacam-se o PMC [4], ADSD [5] e MM [6]. O uso de diagnóstico em grades como estratégia contra ataques maliciosos mostra-se uma solução eficiente, visto que independe das plataformas de hardware e software utilizadas, não requer o uso de esquemas “pesados” de criptografia e permite ser aplicado tanto em grades fechadas (organizações virtuais formadas entre corporações) quanto em grades abertas (constituídas por hosts comuns conectados à Internet e dispostos a doarem seus recursos num esquema de computação voluntária) [7].

3. Um Modelo para Diagnóstico em Grades

O modelo de diagnóstico descrito neste trabalho é utilizado como estratégia para detectar e evitar a presença de nós maliciosos interessados em corromper os resultados das tarefas, garantindo assim a integridade dos processos distribuídos em uma grade. O modelo assume que o diagnóstico é realizado por todos os nós que possuem um *nível mínimo de confiabilidade*, o qual é dado pelo seu histórico comportamental. Assim, são definidas três entidades: nós *executores* (que fornecem recursos), nós *testadores* (que fornecem recursos e testam os nós executores) e nós *Ultra-Confiáveis* (que fornecem recursos e

gerenciam um grupo (cluster) de nós executores e testadores, decidindo, dentre outras atividades, quem é ou não malicioso).

A reputação, inferida através de tarefas de testes (*test-jobs*), determina o status de cada nó. Nós com alta reputação são considerados melhores fornecedores de recursos e por isso tendem a possuir maior status. Nós maliciosos que são detectados trapaceando possuem menor reputação e, portanto, tendem a serem excluídos. A Figura 1 ilustra a estratégia de diagnóstico, onde dois nós do sistema, A e B atuando como testadores, enviam dois test-jobs diferentes para um nó C, atuando como executor (a). O nó C executa a tarefa e devolve os resultados para os respectivos testadores (b). Em seguida, os nós A e B enviam suas percepções sobre o nó C para o nó D, atuando como o UC responsável pelo grupo (c). Com base nessas percepções, o nó D emite um diagnóstico sobre o nó C.

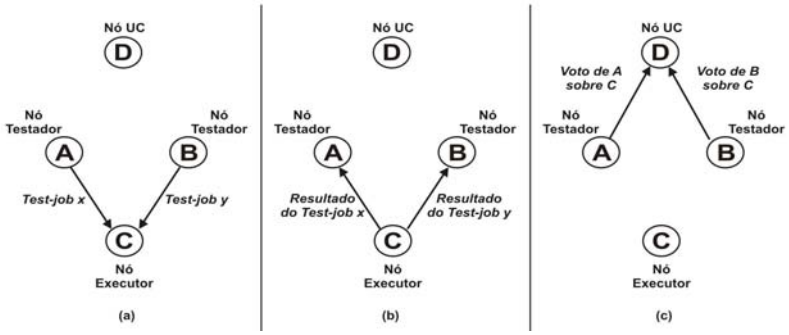

Figura 1. Estratégia de diagnóstico para detecção de nós maliciosos

Se dois nós testadores A e B concordam sobre o estado (malicioso ou idôneo) do nó executor C testado, então o nó D Ultra-Confiável toma aquelas percepções como válidas e, dependendo dos resultados, condena ou não o nó C. Mas se os nós testadores A e B divergem sobre o estado do nó C, então o nó D analisa o histórico comportamental dos testadores, a fim de verificar um padrão de repetição (por exemplo, se um dos nós reprovou todos os testes por ele aplicado, se um dos nós reprovou apenas os testes aplicados sobre aquele nó executor). Baseado nessa análise, o nó UC infere se algum dos nós (testadores ou executor) agiu de forma maliciosa, eliminando-o. Devido à limitação de espaço, os algoritmos do modelo não são aqui apresentados, podendo ser encontrados em [7].

4. Cenários de Simulação e Resultados

Para validação do modelo utilizou-se o simulador de grades GridSim [8]. Foi assumido um ambiente composto de 200 nós, com quotas variadas de nós maliciosos (1/6, 1/3 e 2/3 do total). Estes foram modelados com 25% de chances de retornarem um resultado corrompido. Além disso, foram consideradas diferentes quantidades de rodadas de testes (3, 5, 8, 10, 15 e 20) realizadas em diferentes períodos (a cada 6, 12 e 24h). Após cada rodada, novos nós entram no ambiente.

Dois cenários foram avaliados: no primeiro, os nós executores são gerenciados por um único UC, de modo que os aspectos de reputação não são considerados; no segundo cenário, o modelo é implementado por completo, incluindo a confiabilidade, as métricas temporais, a elevação de status e a reconfiguração de clusters. Por questão de espaço, são apresentados apenas os gráficos relativos ao segundo cenário, com 2/3 dos nós agindo maliciosamente. As métricas observadas foram: a quantidade de nós maliciosos detectados, o custo de processamento introduzido (razão entre o número de test-jobs e o total de jobs) e a acurácia (razão entre o número de jobs processados corretamente e o total de jobs).

A Figura 2 mostra a quantidade de nós maliciosos detectados em função do número de rodadas de testes em diferentes períodos. Como esperado, quanto maior a quantidade de rodadas, maior o número de maliciosos detectados. Nota-se também que a partir de um certo momento, não importa a quantidade de testes, o número de nós maliciosos detectados tende a ser o mesmo. Por exemplo, com a realização de testes a cada 6h, o número de nós maliciosos detectados é praticamente o mesmo a partir de 5 rodadas.

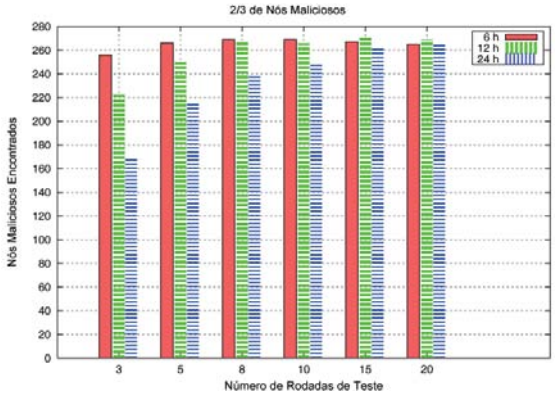

Figura 2. Nós inseridos na blacklist com 2/3 de maliciosos

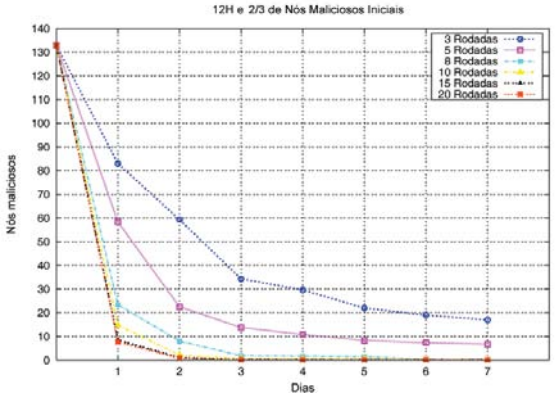

Figura 3. Nós maliciosos remanescentes

O pior caso ocorre quando há uma maior quantidade de nós maliciosos e uma quantidade de testes reduzida. Com apenas 3 rodadas por dia são detectados aproximadamente 170 nós maliciosos. Porém isso não significa que restaram 100 nós com mau comportamento livres para comprometer as aplicações, pois como é considerada a entrada de novos nós após cada rodada, o número de maliciosos no ambiente pode variar. Para avaliar o grau de eficiência do esquema é preciso observar também a quantidade de maliciosos remanescentes. A Figura 3 mostra como o sistema é "higienizado" durante 7 dias de operação, utilizando o algoritmo de diagnóstico proposto em um ambiente com 2/3 da grade inicialmente "contaminada", e testes realizados a cada 12h.

De acordo com os gráficos, para qualquer quantidade de nós maliciosos o sistema mostra-se robusto a partir de 5 rodadas de testes ocorrendo a cada 6h, ou 8 rodadas a cada 12h. Entretanto, para conhecer o número ideal de rodadas e de periodicidade, é necessário observar o custo e a acurácia obtida com estes valores, visto que tais métricas podem impactar nos resultados. O custo introduzido é apresentado na Figura 4, a qual revela que 8 rodadas a cada 12h apresenta-se como melhor alternativa, tendo em vista o trade-off alcançado. Enquanto com 5 rodadas obtém-se um overhead de 17%, com 8 esse overhead cai para 12,3%. Já a Figura 5 apresenta o grau de acurácia obtida com o diagnóstico. A acurácia sofre com um número maior de nós maliciosos no sistema, especialmente se os testes são aplicados com menor freqüência. Enquanto, por exemplo, com 1/6 de maliciosos na grade obtém-se uma acurácia de 99,7% com 5 rodadas a cada 6h, esse valor decresce vertiginosamente para 86,7% com a mesma quantidade de rodadas, acontecendo a cada 24h em uma grade com 2/3 dos nós comprometidos.

Em ambos os cenários, 8 rodadas de teste mostraram-se como uma quantidade ideal para a estratégia de diagnóstico. Enquanto no melhor caso do primeiro cenário pode-se obter um grau de detecção acima de 90%, com uma acurácia de 98% e custo de 15%, no segundo cenário foi observado uma detecção de 100% dos nós maliciosos, com

uma acurácia de 99,7% e um custo de 12,3%, o que demonstra a eficiência do modelo ao detectar e isolar diferentes quantidades de nós maliciosos de uma grade.

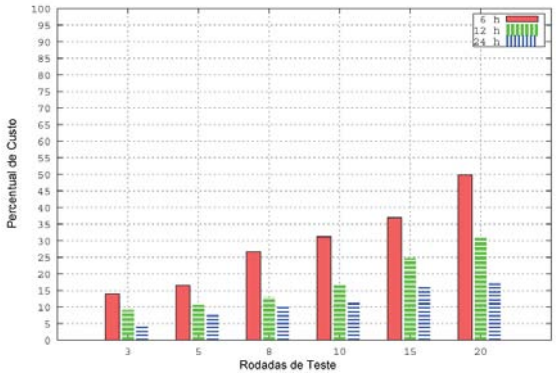

Figura 4. Custo introduzido

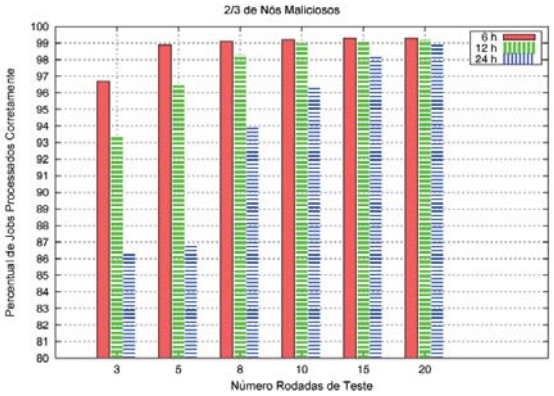

Figura 5. Acurácia

5. Conclusões

A utilização de diagnóstico em nível de sistema como estratégia contra ataques de manipulação de resultados mostra-se uma solução eficaz, visto que independe da plataforma e é interoperável com soluções de segurança locais, o que viabiliza seu emprego na maioria das middlewares de grades. Além disso, o modelo apresentado organiza os nós em clusters lógicos, estabelecendo uma hierarquia entre os mesmos, de acordo com o papel de cada nó. Essa abordagem permite que o diagnóstico seja feito de forma distribuída com a participação dos nós que possuem um nível mínimo de confiabilidade.

Os resultados experimentais confirmaram a robustez e escalabilidade dessa estratégia, uma vez que todos os nós maliciosos podem ser devidamente eliminados do ambiente, obtendo um alto índice de acurácia dos jobs processados com um baixo custo em termos de jobs de teste adicionais. Como trabalho futuro, será implementada uma ferramenta baseada neste modelo de diagnóstico para que a mesma seja incorporada a uma plataforma de grade real, tal como o OurGrid ou Globus.

Referências

[1] A. Setiawan, D. Adiutama, J. Liman, A. Luther, and R. Buyya. Gridcrypt: High performance symmetric key cryptography using enterprise grids. In *PDCAT 2004: Parallel and Distributed Computing - Applications and Technologies*, pages 872–877, 2004.

[2] E. P. Duarte and T. Nanya. A hierarchical adaptive distributed system-level diagnosis algorithm. In *IEEE Transactions on Computers*, volume 47, pages 34–45. IEEE Computer Society, 1998.

[3] M. Hollick. On the effect of node misbehavior in ad hoc networksv. In *Proceedings of IEEE International Conference on Communications (ICC 2004)*, volume 6, pages 3759–3763, 2004.

[4] F. Preparata, G. Metze, and R. Chien. On the connection assignment problem of diagnosable systems. *IEEE Transactions on Electronic Computers*, 16:848–854, 1968.

[5] R. P. Bianchini Jr. and R. W. Buskens. Implementation of online distributed system-level diagnosis theory. *IEEE Transactions on Computers*, 41(5):616–626, 1992.

[6] J. Maeng and M. Malek. A comparison connection assignment for self-diagnosis of multiprocessor systems. In *Digest 11th International Symposium Fault Tolerant Computing*, pages 173–175, 1981.

[7] F. Martins, M. Maia, R. M. de Castro Andrade, A. Luiz dos Santos, and J. Neuman de Souza. Detecting malicious manipulation in grid environments. In *SBAC-PAD - 18th International Symposium on Computer Architecture and High Performance Computing*, pages 28–35. IEEE Computer Society, 2006.

[8] R. Buyya and M. Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. In *Journal of Concurrency and Computation: Practice and Experience (CCPE)*, 2002.