Uma Estratégia para Validação Experimental de um Sistema de Comunicação de Grupo*

Gabriela Jacques da Silva[1†], **Taisy Silva Weber**[1]

[1]Grupo de Pesquisa em Tolerância a Falhas
Programa de Pós-Graduação em Ciência da Computação
Instituto de Informática - Universidade Federal do Rio Grande do Sul
Caixa Postal 15064 - CEP 91591-970 - Porto Alegre - RS
{gjsilva,taisy}@inf.ufrgs.br

Abstract. *This paper presents a strategy to an experimental validation of JGroups group communication system. This kind of system is commonly used as a basic building block for the development of fault-tolerant applications, due to features such as reliable multipoint communication. The validation of this system by fault injection helps to remove errors on the group communication mechanism and to verify the behavior of this system in the presence of faults.*

Resumo. *Este artigo apresenta uma estratégia para a validação experimental do sistema de comunicação de grupo JGroups. Este tipo de sistema geralmente é usado como base para o desenvolvimento de aplicações tolerantes a falhas por apresentar características como, por exemplo, comunicação multiponto confiável. A validação deste sistema por injeção de falhas é essencial na remoção de erros na construção dos mecanismos de comunicação de grupo, como também verificar o comportamento deste na presença de falhas.*

1. Introdução

A medida que sistemas computacionais são incorporados no cotidiano, estes necessitam apresentar características de tolerância a falhas para corresponder a confiança depositada no comportamento correto desses sistemas. Uma das maneiras é a replicação, tanto de *hardware* quanto de *software*. Como técnicas de replicação por *hardware* são bastante caras, foram motivadas várias pesquisas para o desenvolvimento de técnicas de replicação baseadas em *software*.

O uso de múltiplas cópias de uma informação pode gerar problemas de inconsistência, no caso em que as atualizações não sejam feitas na mesma ordem ou então quando estas não são propagadas corretamente para todas as cópias. Comunicação multiponto confiável e ordenamento de mensagens são características oferecidas por sistemas de comunicação de grupo, facilitando portanto a construção de técnicas de replicação [GUERRAOUI e SCHIPER 1996]. Sistemas de comunicação de grupo apresentam vários outros usos clássicos, como suporte para sistemas operacionais distribuídos, transações distribuídas, replicação de bases de dados, balanceamento de carga, gerenciamento de sistemas, servidores de alta disponibilidade e computação colaborativa

*Projeto ACERTE (472084/2003-8)
†Bolsista do Conselho Nacional de Desenvolvimento Científi co e Tecnológico

[CHOCKLER *et al*. 2001]. Devido a esta larga aplicabilidade, faz-se essencial a validação deste tipo deste tipo de sistema.

Este artigo propõe uma estratégia para a validação experimental de um sistema de comunicação de grupo chamado JGroups [BAN 1998]. Para isto são executados experimentos de injeção de falhas para verificar o comportamento deste *middleware* em presença de falhas. A próxima seção aborda a importância da validação experimental e também a técnica de injeção de falhas. A seção 3 descreve características gerais dos sistemas de comunicação grupo e também JGroups. A seção 4 apresenta a estratégia para validação deste sistema, explicitando os modelos de falhas e o injetor de falhas em uso. A última seção apresenta algumas conclusões e trabalhos futuros.

2. Validação Experimental

Uma fase fundamental no desenvolvimento de sistemas tolerantes a falhas é a fase de validação. Delegar a verificação do funcionamento do mecanismo de tolerância a falhas para uma situação de uso efetivo do *software* (uma falha real) pode gerar conseqüências completamente desastrosas. Para verificar o funcionamento correto destes mecanismos, os sistemas devem ser validados. A validação pode ser analítica e experimental, sendo estas duas formas complementares. Uma das técnicas usadas para validar experimentalmente um sistema é através da injeção de falhas. A injeção de falhas testa a eficiência dos mecanismos de tolerância a falhas e avalia a segurança de funcionamento dos sistemas, provendo uma realimentação no processo de desenvolvimento [IYER 1995].

Um ambiente de injeção de falhas é formado geralmente por um sistema alvo, um injetor de falhas, uma biblioteca de falhas, um gerador de carga de trabalho (*workload*), uma biblioteca de carga de trabalho, um controlador, um monitor e um coletor e analisador de dados [HSUEH *et al*. 1997]. O funcionamento básico destes ambientes parte da injeção de falhas no sistema alvo com o injetor de falhas. O sistema alvo é alimentado com o gerador de carga de trabalho. A execução do sistema é monitorada pelo monitor que comunica eventos para o coletor de dados. Este coletor rastreia a execução, que pode ser posteriormente analisada pelo analisador de dados.

Ferramentas de injeção de falhas podem ser desenvolvidas tanto em *hardware* quanto em *software*. A decisão entre uma implementação e outra depende não só do tipo de falha que se quer injetar mas também do esforço despendido para a criação de cada uma. O uso de um injetor de falhas implementado em *software* apresenta um menor custo, já que não é necessário um dispositivo específico. Além disso, é mais simples injetar falhas em camadas superiores, como o sistema operacional e a própria aplicação em teste. A estratégia apresentada compreende apenas injetores de falhas implementados em *software*. A seção a seguir trata sobre sistemas de comunicação de grupo e também descreve brevemente o sistema JGroups, que é o sistema alvo da validação.

3. Sistemas de Comunicação de Grupo

Sistemas de comunicação de grupo são *middlewares* para comunicação multiponto confiável. Os membros de um grupo podem não ser apenas processos, mas também objetos, ou ainda qualquer entidade que possa enviar e receber mensagem para/de um grupo [BAN 1998]. Estes *middlewares* também oferecem propriedades como acordo, validade,

integridade e terminação. Outra possibilidade é o ordenamento no recebimento de mensagens, característica necessária em sistemas que necessitam que atualizações de valores sejam realizadas de forma consistente.

Outro serviço geralmente oferecido por estes sistemas é o serviço de *membership*. Este serviço é responsável por gerenciar os membros de cada grupo a cada instante, que pode mudar sempre que um membro se junta ou abandona um grupo (voluntariamente ou em casos de falha). Para representar a cada momento os membros de um grupo é usado o conceito de visão. Os membros de um grupo são comunicados da adesão ou do abandono de um membro através da instalação de uma nova visão.

Através das várias características e serviços oferecidos por um sistema de comunicação de grupo, estes se tornam uma peça básica para a construção de sistemas distribuídos tolerantes a falhas. Um sistema que está sendo usado atualmente em vários projetos é JGroups, que será brevemente descrito na próxima seção.

3.1. JGroups

JGroups [BAN 1998] é um *toolkit* para comunicação de grupo confiável desenvolvido em Java. A abstração de grupos é feita através de um canal. Todos os membros que realizarem a conexão em canais de mesmo nome fazem parte de um mesmo grupo. Quando um canal é criado pelo usuário, pode ser escolhida a pilha de micro-protocolos que este canal irá usar. Desta forma o usuário tem a liberdade de escolher quais as restrições que o canal deve impor sobre os membros deste, como por exemplo o ordenamento de mensagens. Outro objetivo deste sistema é oferecer abstrações de mais alto nível para a comunicação de grupo, facilitando o uso deste paradigma. Para isso, JGroups oferece vários blocos de *software* prontos com padrões de comunicação de grupo implementados.

Atualmente vários projetos usam JGroups como um bloco auxiliar para prover alta disponibilidade. Entre eles pode citar-se o JBoss [JBOSS 2003], que é um servidor de aplicações J2EE gratuito e usa JGroups para a implementação da clusterização de servidores. Outro sistema é Dorothy [PASIN *et al*. 2002]. JGroups neste sistema serve como base para a realização da consistência de réplicas de componentes Enterprise JavaBeans.

Na seção a seguir é apresentada uma estratégia para validar experimentalmente este sistema de comunicação de grupo. Esta validação implica em vários benefícios, como a remoção de erros latentes no mecanismo de comunicação de grupo e também a verificação do comportamento deste em caso de falhas. Realizar experimentos de validação em JGroups auxilia não só na validação deste sistema em si, mas também para a validação das aplicações que o usam como base para alcançar alta disponibilidade.

4. Estratégia para Validação de JGroups

A condução de um experimento de injeção de falhas envolve várias decisões, como o modelo de falhas a ser considerado, a ferramenta injetora de falhas, as aplicações de geração de carga de trabalho, cenários de testes e as métricas para avaliação do sistema alvo.

4.1. Modelo de Falhas

Uma falha ocorre sempre que um serviço não é prestado de acordo com a sua especificação. Dependendo do tipo desta falha ela recebe uma classificação. Como um sistema de comunicação de grupo é executado em um contexto distribuído, o modelo de

falhas que será considerado para este experimento será o modelo de falhas para sistemas distribuídos definido por Cristian [CRISTIAN 1991].

Este modelo descreve falhas de colapso, omissão, temporização e valor. Uma falha por colapso ocorre quando o servidor pára totalmente de responder. Este tipo de falha ainda pode ser classificada em subtipos, levando em consideração o estado do servidor após a sua recuperação. Uma falha de omissão ocorre quando um servidor não responde a uma requisição. Uma falha é considerada de temporização quando uma resposta do servidor ocorre fora do intervalo de tempo especificado, podendo ser tanto uma resposta cedo demais quanto tardia demais. A última geralmente é associada a falhas de desempenho. A falha por valor ocorre quando o valor de retorno de uma requisição é incorreto. Este tipo de falha não será considerada para este experimento de validação. O sistema alvo, o JGroups, está especificado para tolerar apenas falhas de colapso. A estratégia descrita se resume a este modelo, porém pode ser estendida a outros sistemas que possuem modelos de falhas menos restritivos que apenas o de colapso.

4.2. Ferramenta de Injeção de Falhas

O modelo de falhas considerado inclui falhas de omissão, temporização e colapso. Para provocar falhas de omissão e temporização é necessário provocar falhas no envio de mensagens. Com isso, a falha deve ser injetada no sistema de troca de mensagens. Um meio de provocar falhas por colapso é fazer com que os outros membros do grupo não recebam mais mensagens do membro falho. Provocar o corte completo do sistema de comunicação de um membro do grupo provocará a percepção de uma falha por colapso nos outros membros do grupo, já que não será possível trocar mensagens com este membro em colapso.

Para injetar falhas de comunicação foi escolhido o injetor de falhas GOOFI (*Generic Object-Oriented Fault Injection*) [AIDEMARK *et al*. 2001]. Esta ferramenta é genérica, já que não é presa a nenhuma técnica específica de injeção de falhas. Desta maneira é construído um ambiente de fácil adaptação para injeção de falhas necessárias para um determinado sistema alvo. Esta ferramenta é altamente portável por ser desenvolvida em Java e usar um banco de dados compatível com a linguagem SQL. A arquitetura desta ferramenta pode ser vista na figura 1.

A arquitetura da ferramenta pode ser divida em três camadas. A GUI (*Graphical User Interface*) é usada para configurar e iniciar os experimentos de injeção de falhas. A camada central inclui as classes Java `FaultInjectionAlgorithms`, `Framework`, `TargetSystemInterface`. Estas classes definem métodos abstratos que devem ser especificados de acordo com o algoritmo de injeção de falhas e o sistema alvo do experimento. Neste trabalho estas classes serão estendidas para gerar falhas de comunicação e para interagir com JGroups. A última camada da arquitetura é a interface com o banco de dados. Esta é usada para armazenar informações sobre o sistema alvo, os experimentos de injeção de falhas e os dados capturados de cada experimento.

Um modo complementar proposto para injetar falhas no sistema JGroups é através da extensão e criação de alguns micro-protocolos, que podem ser configurados no momento da criação de um canal. JGroups dispõe de dois protocolos para atraso e descarte de mensagens. Estes podem ser estendidos para adequá-los aos experimentos. O micro-protocolo de atraso pode ser usado para injetar falhas de temporização e o de descarte para falhas de omissão. Para injeção de falhas de colapso, todas as mensagens de

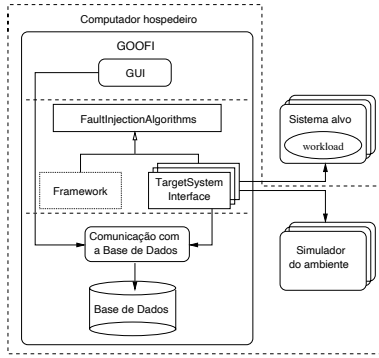

Figura 1: Arquitetura da ferramenta GOOFI

todos os canais de comunicação do emissor devem ser descartadas após a primeira omissão.

4.3. Geração de Carga de Trabalho

Para gerar carga de trabalho são executadas aplicações que usam serviços essenciais ou freqüentemente oferecidos por um sistema de comunicação de grupo. Um serviço que deve ser bastante explorado é o de *membership*, já que várias aplicações baseiam-se neste serviço para prover alta disponibilidade. Outros serviços que devem ser testados são o de ordenação total, causal e FIFO. Um sistema que usa intensivamente ambos serviços é Dorothy e está sendo utilizado como carga de trabalho.

4.4. Cenários de Testes

A carga de trabalho usa o *middleware* de comunicação de grupo para manutenção de consistência de réplicas. Para a condução de experimentos em ambientes com objetos replicados são necessários no mínimo cinco servidores. Um cenário adequado é injetar falhas em um único servidor que contenha uma das réplicas e monitorar o comportamento dos outros servidores (membros do grupo de replicação). O efeito esperado de uma injeção de falha de colapso em um membro é a troca de visão do grupo nos membros não-falhos. Para cada falha de colapso provocada deve corresponder uma troca de visão, refletindo a exclusão do membro falho do grupo.

Atualmente estão sendo feitos testes de injeção de falhas em um único servidor. Esta abordagem centralizada tem a vantagem de facilitar o controle do experimento, porém reduz os experimentos para somente falhas simples (um único servidor por vez). Posteriormente serão feitos testes para provocar falhas de colapso em dois ou mais servidores simultaneamente ou ainda falhas em cascata.

4.5. Métricas de Avaliação

A primeira métrica que deve ser obtida é a cobertura de falhas do detector de falhas do JGroups. Esta medida pode ser tomada de forma indireta computando o número de mudanças de visão devido ao colapso do servidor. Dividindo-se este número pelo número de falhas injetadas no experimento chega-se a cobertura de falhas. Esta medida permite uma realimentação no desenvolvimento do mecanismo de detecção de falhas do JGroups.

As duas outras métricas de avaliação são as perdas de desempenho provocadas por falhas e a disponibilidade do sistema. Estas medidas são importantes para o desenvolvedor de aplicações tolerantes a falhas que se baseiam neste sistema de comunicação de grupo. Um modo de obtenção da queda de desempenho devido a presença de uma falha é através da comparação do tempo médio de resposta de um cliente do grupo de replicação em duas situações. A primeira situação é quando o serviço está livre de falhas e a segunda é em presença de falhas. A carga de trabalho deve ser a mesma em ambas as situações.

Para obter a medida de disponibilidade do sistema é necessário verificar o tempo de recuperação do sistema após a ocorrência de uma falha. Inicialmente esta métrica pode ser medida calculando a diferença entre o tempo de resposta ao cliente com e sem a presença de falhas no servidor. Este tempo adicional é considerado o tempo de indisponibilidade ou de recuperação.

5. Conclusão

A validação experimental de sistemas de comunicação de grupo é essencial, já que freqüentemente estes são usados como um bloco de suporte a aplicações tolerantes a falhas. A validação destes sistemas traz benefícios não somente para o próprio sistema, mas também para as aplicações que o usam como *middleware* de comunicação de grupo.

O artigo apresentou uma estratégia para conduzir experimentos de validação usando o injetor de falhas GOOFI no sistema alvo JGroups. Uma extensão desta estratégia será usada, após a validação de JGroups, para validar aplicações de alta disponibilidade baseadas em replicação de componentes, como providas pelo sistema Dorothy que se encontra em fase final de implementação.

Referências

AIDEMARK, J.; VINTER, J.; FOLKESSON, P.; KARLSSON, J. *GOOFI: Generic Object-Oriented Fault Injection Tool*. In Proceedings of International Conference on Dependable Systems and Networks 2001. Gotemburgo, Suécia. Julho 2001.

BAN, B. *JavaGroups - Group Communication Patterns in Java*. Department of Computer Science, Cornell University. Julho 1998.

CHOCKLER, G. V.; KEIDAR, I.; VITENBERG, R. *Group Communication Specifications: A Comprehensive Study*. ACM Computing Surveys, Vol. 33, No. 4, pp. 427-469. Dezembro 2001.

CRISTIAN, F. *Understanding Fault-Tolerant Distributed Systems*. Communications of the ACM, Vol. 34, No. 2, pp. 56-78. Fevereiro 1991.

GUERRAOUI, R.; SCHIPER, A. *Fault-Tolerance by Replication in Distributed Systems*. In International Conference on Reliable Software Technologies. Springer Verlag, Lecture Notes in Computer Science 1088. 1996.

HSUEH, M.; TSAI, T. K.; IYER, R. K. *Fault Injection Techniques and Tools*. Computer, Vol. 30, No. 4, pp. 75-82. Abril 1997.

IYER, R. K. *Experimental Evaluation*. In Proceedings of 25^{th} International Symposium on Fault-Tolerant Computing. Pasadena, Estados Unidos. Junho 1995.

JBoss J2EE Application Server. `http://www.jboss.org`

PASIN, M.; WEBER, T. S.; RIVEILL, M. *A Multi-layer Architecture for High Available Enterprise JavaBeans*. Anais do III Workshop em Tolerância a Falhas. Búzios, Rio de Janeiro. Maio 2002.