Tolerância a falhas no modelo MultiCluster

Marcos Locateli Gontarski, Marcos Ennes Barreto

Centro Universitário La Salle (UNILASALLE)
Av. Victor Barreto 2288, CEP: 92.010-000 – Canoas – RS – Brazil
locateli@bancomatone.com.br; barreto@unilasalle.edu.br

Resumo. *Técnicas de tolerância a falhas constituem uma das principais abordagens para o provimento de alta disponibilidade para aplicações críticas que executam em agregados. Este artigo descreve o serviço de tolerância a falhas projetado para a biblioteca DECK, usada como ambiente de execução do modelo MultiCluster.*

1. Introdução

Arquiteturas de agregados têm sido usadas já há algum tempo como uma alternativa de alto desempenho e relativo baixo custo para a execução de aplicações paralelas que exigem grande poder de processamento [Buyya 1999], e vêm sendo empregadas atualmente nos mais diversos cenários, desde centros de pesquisa aplicada até grandes empresas de processamento e armazenamento de dados.

No cenário comercial, arquiteturas de agregados têm sido usadas para diferentes tipos de servidores (aplicação, armazenamento, banco de dados, entre outros). O uso crescente deste tipo de arquitetura se deve principalmente ao relativo baixo custo que ela apresenta e ao emprego de redes de comunicação de alto desempenho; pontos que tornam estas arquiteturas bastante propícias ao desenvolvimento e à execução de aplicações críticas, tais como sistemas de comércio eletrônico e sistemas bancários. Nesse contexto, o uso de técnicas de tolerância a falhas é um dos principais mecanismos para a obtenção de alta disponibilidade e confiabilidade na execução dessas aplicações [Hwang 1998].

No lado acadêmico, as arquiteturas de agregados têm sido usadas para o desenvolvimento de técnicas de replicação de dados, escalonamento dinâmico e migração de processos, simulação de redes de sensores, entre outros [Youn et al 2000].

Este trabalho descreve o serviço de tolerância a falhas para a biblioteca de programação DECK, usada como ambiente de execução do modelo MultiCluster [Barreto et al 2000]; e apresenta também uma validação preliminar de algumas funcionalidades deste serviço.

2. Agregados de alta disponibilidade

Histórica e idealmente, agregados têm sido mais usados para alcançar alto desempenho do que alta disponibilidade. Em um agregado de alto desempenho, os processos executam paralelamente em vários nós (geralmente) homogêneos, trabalhando de forma cooperativa e compartilhando recursos de processamento e de entrada e saída. A homogeneidade e a baixa latência de comunicação, aliadas ao baixo custo

incremental, tornam os agregados uma excelente plataforma para a execução de aplicações que demandam por alto poder de processamento.

Nos agregados de alta disponibilidade, os nós compartilham os mesmos dispositivos de armazenamento, substituindo-se uns aos outros em caso de falha de hardware ou software. Em comparação ao caso anterior, num agregado de alta disponibilidade, nem todos os nós estão diretamente disponíveis ao usuário – determinados nós são mantidos como reservas, com funções de armazenamento e replicação de dados e programas; e estão aptos a assumir o processamento em caso de falha em um nó principal.

Agregados de alta disponibilidade normalmente não compartilham a carga de processamento como os agregados de alto desempenho, nem distribuem tráfego como os agregados que fazem balanceamento de carga. Nos agregados de alta disponibilidade, além da replicação dos dados, o serviço de tolerância a falhas pode migrar tarefas de um nó defeituoso para um nó operacional e reiniciar aplicações [Buyya 1999].

O software usado em agregados de alta disponibilidade geralmente agrupa os recursos que devem ser monitorados e, em caso de falha, reinicializados em outro local. Este "grupo de recursos" pode então ser gerenciado através de alguma técnica de tolerância a falhas [Youn et al 2000]:

- *Idle standby*: um nó reserva é capaz de assumir a gerência do grupo de recursos em caso de falha do nó principal. Pode haver interrupção de serviços caso o nó principal se recupere da falha e reassuma o grupo de recursos;
- *Rotating standby*: técnica semelhante a anterior, porém o nó principal, ao retornar do estado falho, se mantém como nó reserva do "novo" nó gerenciador. Neste caso, a interrupção de serviços pode ser minimizada ou eliminada;
- *Simple failover*: o nó principal executa uma aplicação crítica, enquanto o nó reserva executa uma aplicação não-crítica e mais os serviços de monitoramento do nó principal. No caso de falha do nó principal, o reserva pára a aplicação não-critica e assume a aplicação crítica;
- *Mutual takeover*: esquema onde dois ou mais nós são igualmente capazes de assumir o grupo de recursos, um do outro. Pode ser usado em aplicações fracamente acopladas, onde cada grupo de nós executa um conjunto de tarefas distintas e, em caso de falhas, outro nó ou grupo de nós é capaz de assumir a execução das tarefas; e
- *Concurrent access*: esquema onde os nós acessam os mesmos recursos (um dispositivo de armazenamento externo, por exemplo) de forma concorrente, de modo que os dados produzidos em um nó possam ser recuperados por outro nó em caso de falhas.

3. Tolerância a falhas no modelo MultiCluster

O MultiCluster corresponde a um modelo para a concepção de grades locais a partir da integração de agregados de alto desempenho. O modelo utiliza a biblioteca DECK sobre a implementação em linguagem C dos protocolos de gerência e comunicação definidos no projeto JXTA [JXTA 2005].

DECK é uma biblioteca de programação desenvolvida no Instituto de Informática da UFRGS desde 1998; e que contém atualmente diferentes versões para diversas tecnologias (Ethernet, Myrinet, SCI e Infiniband) e protocolos (TCP, BIP, GM, SISCI, VIA) de comunicação. Todas as implementações DECK são organizadas em duas camadas: a camada inferior fornece recursos de multiprogramação, comunicação e conexão entre agregados; e a camada superior é responsável pelo fornecimento de serviços especializados (comunicação coletiva, tolerância a falhas, roteamento de mensagens, etc.).

No MultiCluster, diferentes cenários de utilização de grades são considerados (Figura 1), onde em cada cenário o usuário define parâmetros para a integração e comunicação entre os recursos integrados, através de arquivos de configuração.

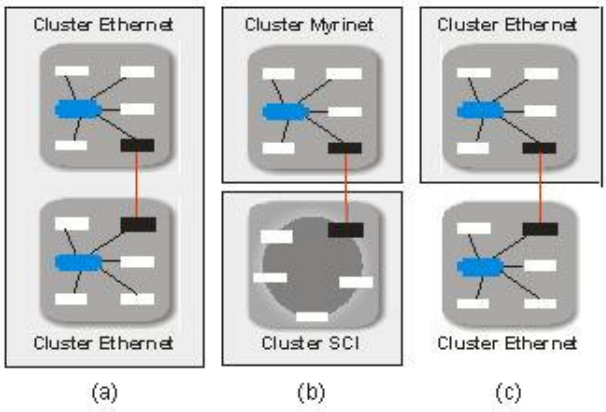

Figura 1. Cenários de integração do modelo MultiCluster.

Em cada agregado, um nó é configurado como máquina de acesso (marcada em preto na figura). Neste nó são executados alguns serviços especializados que permitem o controle dos demais nós do agregado, bem como o roteamento de mensagens entre nós pertencentes a diferentes agregados

No cenário *a* o objetivo da integração é prover escalabilidade para aplicações fortemente acopladas, onde a aplicação considera a existência de nós com comunicação direta entre si, independente de suas localizações físicas; sendo distribuída de maneira uniforme entre os nós (segundo o modelo SPMD).

No cenário *b* o objetivo da integração é fazer uso seletivo dos agregados, de modo que a aplicação seja distribuída de acordo com as necessidades de cada tarefa e das características providas em cada recurso. Neste caso, os agregados são vistos como recursos distintos e as operações de comunicação são realizadas somente entre tarefas que pertencem a um mesmo agregado.

O cenário expresso na letra *c* corresponde a uma integração voltada para tolerância a falhas, que pode ser usada em combinação com qualquer um dos cenários anteriores. Neste cenário, a aplicação executa em um agregado, mas o modelo prevê outro conjunto de nós que é usado em caso de falhas. A máquina de acesso do agregado principal mantém uma conexão com a máquina de acesso reserva. No caso de falha de um dos nós, a tarefa atribuída ao mesmo pode ser repassada ao agregado reserva para que seja re-executada (Figura 2a e 2b). No caso de falha da máquina de acesso do agregado principal, os nós deste agregado podem “eleger” a máquina de acesso do

agregado reserva para o papel de controlador, passando a esta máquina o resultado das suas execuções (Figura 2c e 2d).

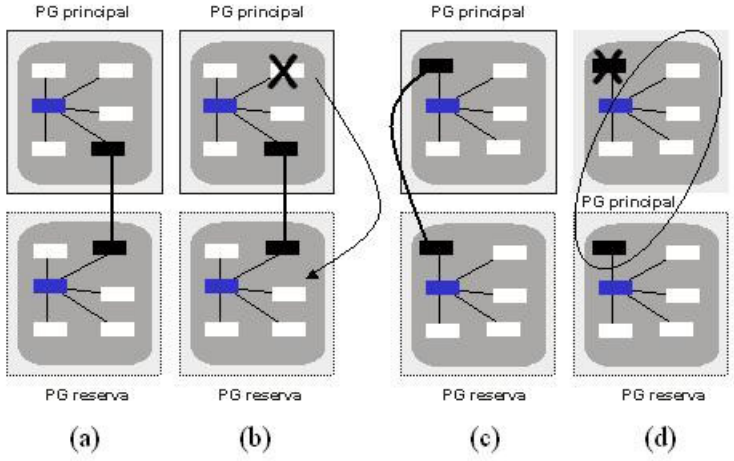

Figura 2. Migração de tarefa (2 a e 2b) e de *peergroup* (2c e 2d) no MultiCluster.

No serviço proposto, um agregado é definido como *peergroup* (PG) principal; e nele, a máquina de acesso executa os serviços de tolerância a falhas, gerência do agregado e roteamento de mensagens. Estes dois últimos serviços configuram esta máquina como um *rendezvous peer* e um *relay peer*, que na semântica do serviço, representam um nó que armazena informações sobre os demais pares e um nó usado para comunicação com pares pertencentes a outros *peergroups*, respectivamente. Adicionalmente, outro agregado é especificado como *peergroup* reserva, contendo um determinado número de nós e uma máquina de acesso; que executa os mesmos serviços, porém em caráter de monitoramento.

O serviço de tolerância a falhas proposto possui as seguintes características:

- implementa uma técnica mista (*idle standby* + *simple failover*): se o controlador do PG principal falha, o controlador reserva assume o controle do grupo até o final da execução da aplicação, mesmo que adiante o controlador principal retorne do estado falho. Além disso, o controlador do PG reserva executa funções de monitoramento do PG principal;
- a detecção de falhas dá-se mediante falhas de comunicação entre quaisquer pares da rede, e depende do tipo de aplicação que está sendo executada no MultiCluster. Ao escolher o cenário de tolerância a falhas, toda comunicação entre os pares obedece a um tempo máximo (*timeout*) de resposta. Se a aplicação for fortemente acoplada, a falha em um nó faz com que o serviço migre esta tarefa para um nó reserva, restaurando o contexto da tarefa a partir de informações gravadas em um arquivo de log a cada troca de mensagens. Se a aplicação for fracamente acoplada, então o serviço simplesmente aloca um nó reserva para que a tarefa executada novamente.
- A falha do controlador do PG principal é detectada quando algum par deixa de receber mensagens de monitoramento do controlador. Neste caso, este par estabelece uma conexão com o controlador reserva, passando a reportar a este coordenador as suas informações de monitoramento.

Tanto a migração de tarefas quanto a migração de *peergroup* somente é possível devido a possibilidade de configuração de *rendezvous peers* e *relay peers* dentro do modelo MultiCluster. Além disso, outro protocolo JXTA é essencial para o desenvolvimento do serviço: o *Peer Info Service*. Através deste serviço, um nó pode receber informações (carga de trabalho, estado de execução, etc.) dos demais nós. O serviço de tolerância a falhas usa este protocolo para monitorar o estado de cada nó (par) pertencente ao *peergroup*.

4. Validação do serviço

Parte do serviço de tolerância a falhas proposto foi implementado e está em fase de validação. O protótipo inicial foi desenvolvido em uma rede Linux, distribuição Fedora (2.6.5-1), com a implementação DECK/TCP e JXTA-C versão 2.1.

Neste protótipo, dois *peergroups* (principal e reserva) foram estabelecidos, cada qual com 4 pares (controlador + 3 nós). As aplicações usadas para a validação são:

- Replicação de arquivos entre pares de um *peergroup* e simulação de falha em um determinado par, o que ocasiona a transferência do arquivo replicado para um par pertencente ao *peergroup* reserva;
- Multiplicação de matrizes, onde um determinado par falha após processar uma parte da aplicação, fazendo com que a tarefa e o seu contexto tenham que ser migrados para um nó do *peergroup* reserva; e
- Ordenação distribuída de vetores, onde cada nó recebe uma parte do vetor a ser ordenado. Quando um par falha, a parte do vetor deve ser passada para um nó do *peergroup* reserva.

A figura 3 apresenta os resultados obtidos com a replicação de arquivos, executando em uma rede com máquinas Pentium III de 1.8 GHz e 256 MB de memória.

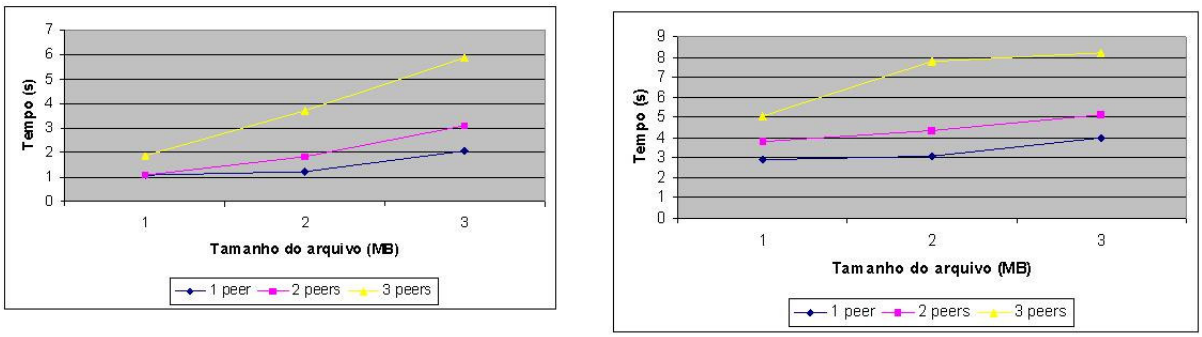

Figura 3. Tempo de replicação de arquivos no protótipo.

Nesta simulação, a quantidade de pares no *peergroup* principal e o tamanho do arquivo replicado foram variados. Na figura à esquerda são apresentados os tempos de replicação para um *peergroup* com 1, 2 e 3 nós processadores, além do nó controlador. Na figura à direita são apresentados os tempos de replicação para os mesmos casos, considerando a falha de um dos nós e conseqüente migração do arquivo replicado para um nó do *peergroup* reserva.

5. Trabalhos relacionados

Existem na literatura diversas soluções de alta disponibilidade e confiabilidade para agregados e servidores Web. Em [Youn et al 2000], um conjunto de propostas baseadas em sistemas Unix e Linux são comparadas em termos das técnicas de hardware e software utilizadas para o fornecimento de tolerância a falhas.

Em [Anwar et al 2004], um conjunto de protocolos de comunicação em grupo usados em redes P2P é analisado com vistas ao emprego deste tipo de protocolo em aplicações militares. A análise destes sistemas revela que redes P2P ainda provêem poucas funcionalidades relacionadas a segurança e tolerância a falhas.

Dentro do projeto JXTA existem algumas iniciativas relacionadas com tolerância a falhas: o projeto do (*distributed objects*) usa os protocolos JXTA para a replicação de objetos; e o projeto JXTA-RM implementa um protocolo de *multicast* confiável.

6. Conclusões

O fornecimento de técnicas de tolerância a falhas em redes P2P ainda é uma área nova de pesquisa, com poucas propostas já divulgadas. Neste trabalho, um serviço de tolerância a falhas proposto para o modelo MultiCluster é descrito. Este serviço está baseado na utilização de diferentes *peergroups*, que são usados para a execução da aplicação e de tarefas de monitoramento e tratamento de falhas, em diferentes cenários de utilização do modelo MultiCluster.

Referências bibliográficas

Hee Yong Youn, Chansu Yu, Dong Soo Han, and Dongman Lee. (2000) The Approaches for High Available and Fault-Tolerant Cluster Systems." In.: Workshop on Fault Tolerant Control and Computing (FTCC-1), pp.107-116, May 2000.

JXTA (2005). Projeto JXTA. http://www.jxta.org.

Kai Hwang. (1998) Scalable parallel computing. New York: McGraw-Hill, USA.

Marcos Barreto, Rafael Ávila e Philippe Navaux (2000) The MultiCluster model to the integrated use of multiple workstation clusters. In.: International Workshop on Personal Computers based Networks of Workstations (PC-NOW 2000). Quintana Roo, México.

Pankaj Jalote (2002) Fault tolerance in distributed systems. New Jersey: Prentice Hall, USA.

Rajkumar Buyya (ed) (1999) High performance cluster computing – architectures and systems. Vol. 1. New Jersey: Prentice Hall, USA.

Zahid Anwar, William Yurcik, Roy Champbell. (2004) A survey and comparison of peer-to-peer group communication systems suitable for network-centric warfare. Disponível em http://www.ncsa.uiuc.edu (Novembro de 2004).