Em Direção a um Serviço de Replicação Transparente para Aplicações Web na Arquitetura J2EE[1]

André Zampieri, Luciano Azevedo Cassol

Departamento de Informática - Universidade de Caxias do Sul (UCS)
Rua Francisco Getúlio Vargas, 1130 - 95070-560 - Caxias do Sul – RS - Brasil

azampier@ucs.br, lacassol@ucs.br

Abstract. *Due to the increasing complexity of computational systems, the use of development platforms has become very popular. They provide many services that allow developers to keep their focus of attention in the applications they are implementing. The J2EE platform is the technological solution by Sun Microsystems that provides support services for the development of Web applications. However, the J2EE platform doesn't offer support to applications that have fault tolerance as a requisite. This work presents the use of Aspect-Oriented Programming to add transparently an objects replication service for applications developed under the J2EE platform.*

1. Introdução

A sociedade moderna depende fortemente dos serviços prestados por sistemas computacionais. A informática vêm sendo usada de forma intensa nas várias atividades humanas. A Internet se tornou a plataforma chave para serviços de entretenimento e de informação, incluindo-se aí sistemas colaborativos, comércio eletrônico, serviços bancários, relacionamento com clientes e parceiros de negócio entre outras. Este cenário evidencia a exigência de sistemas computacionais robustos, os quais possuem um maior grau de complexidade na sua criação pois devem implementar requisitos de segurança, disponibilidade, confiabilidade, desempenho, etc.

Para lidar com essa complexidade, a tendência atual do desenvolvimento de sistemas computacionais é fazer uso de arquiteturas baseadas em componentes. Esse modelo permite uma maior modularização do software, facilitando a reutilização de componentes e simplificando o desenvolvimento pois o ambiente de execução implementa diversos serviços complexos necessários para a aplicação. A especificação J2EE (*Java 2 Enterprise Edition*) [Kurniawan 2002] é a solução tecnológica provida pela SUN Microsystems que, em conjunto com outras especificações, fornece uma abordagem baseada em componentes para o desenvolvimento de aplicações transacionais distribuídas.

Os serviços da plataforma J2EE não foram especificados para a criação de aplicações com requisitos de confiabilidade e disponibilidade. Assim, mecanismos de tolerância a falhas devem ser implementados pela própria aplicação. Isso exige que os projetistas e desenvolvedores de software, além de lidar com os problemas pertinentes das suas aplicações, devem desenvolver e inserir estes mecanismos nos seus códigos, aumentando o tempo de desenvolvimento e conseqüentemente o custo destas aplicações.

[1]Este trabalho é financiado pelo CNPq (processo 506374/2004-1)

Neste artigo é mostrada uma forma de inserir objetos de uma aplicação, que necessitam ser tolerantes a falhas, em um cache de objetos através do uso da Programação Orientada a Aspectos (POA) [Filman et al. 2005]. O cache de objetos pode ser configurado para ser replicado em diversas JVM (*Java Virtual Machine*), fazendo com que os objetos nele inseridos sejam replicados em todas as instâncias da JVM. O que difere este trabalho com o de [Penchikala 2004] é a disponibilização de um *framework* que oferece estruturas de componentes e aspectos para permitir a inserção de replicação em aplicações corporativas no ambiente web.

O restante deste trabalho está organizado da seguinte forma. A seção 2 descreve a replicação através de um cache de objetos. A seção 3 justifica o uso da Programação Orientada a Aspectos e descreve como utilizar aspectos para inserir o mecanismo de replicação em uma aplicação J2EE. A seção 4 conclui o artigo.

2. Replicação Através de um Cache de Objetos

Um cache de objetos é um mecanismo que permite armazenar em memória dados freqüentemente acessados, reduzindo acessos ao armazenamento secundário e com isso aumentando o desempenho das aplicações [Penchikala 2004].

Um dos módulos componentes da família de serviços do servidor de aplicações JBoss é o JBossCache [JBossCache Web Site]. Este módulo disponibiliza duas API´s que implementam um cache de objetos, o TreeCache e o TreeCacheAop . Estas API´s permitem estabelecer um cache que pode ser utilizado de forma local, sem replicação, ou distribuído, com replicação síncrona ou assíncrona (configurada pelo programador).

O TreeCache implementa uma estrutura em árvore, com um conjunto de dados, na forma de uma tabela HASH (hashmap) de chaves e valores em cada nodo [Ban et. al]. Nos nodos da árvore podem ser inseridos objetos POJO[2]. Esta árvore pode ser configurada para ser local ou replicada. As árvores locais existem somente na máquina virtual na qual elas foram criadas e não replicam as alterações para os outros nodos do grupo. As árvores replicadas propagam qualquer mudança nos seus nodos para todas as árvores replicadas no mesmo grupo. Um grupo pode incluir diferentes computadores em uma rede ou diferentes máquinas virtuais java em um único computador. Para realizar a troca confiável de mensagens entre os membros do grupo que possuem réplicas das árvores é utilizado o sistema de comunicação de grupos JGroups [JGroups Web Site].

O TreeCacheAop herda todas as funcionalidades disponibilizadas pelo TreeCache, acrescentando a capacidade de, dinamicamente, através da POA, prover a capacidade de gerenciamento transparente dos objetos inseridos nos seus nodos. Isto é obtido pelo uso dos métodos *get/set* associados com os POJOs.

3. Replicação Através da Programação Orientada a Aspectos

Um sistema computacional possui requisitos que podem ser classificados em funcionais e não-funcionais. A programação orientada a objetos permite uma boa modularização dos requisitos funcionais, porém, não oferece módulos adequados para a implementação dos não-funcionais. A implementação destes resulta em espalhamento de código (um requisito está presente em vários módulos do sistema) e intrusão (quando módulos de

[2] Objetos Java que seguem a estrutura de construtor padrão sem argumentos e métodos acessadores (*getters* e *setters)* para seus atributos.

um sistema devem, simultaneamente, interagir com diversos requisitos) [Lippert and Lopes 2000]. Os requisitos que se espalham em diversos módulos do sistema computacional são denominados de requisitos transversais. A combinação do espalhamento de código com a intrusão leva ao desenvolvimento de sistemas com baixa rastreabilidade, baixa produtividade e baixa reusabilidade do código, resultando em código com baixa qualidade e de difícil evolução.

A inserção do mecanismo de replicação em uma aplicação, através da criação de objetos das classes TreeCache e TreeCacheAop, faz com que surjam os problemas citados acima. Estes problemas (requisitos transversais) podem ser evitados com o uso da Programação Orientada a Aspectos [Filman 2005][Kiczales et al. 2001][Kiczales et al. 1997]. Ela complementa a Programação Orientada a Objetos, onde os requisitos funcionais do sistema são implementados por classes e os requisitos transversais do sistema são implementados por módulos denominados Aspectos. Os aspectos modificam classes, métodos e/ou outros aspectos através de mecanismos estáticos e mecanismos dinâmicos. Os mecanismos estáticos preocupam-se com a adição de estado e comportamento nas classes, enquanto que os mecanismos dinâmicos modificam a semântica destas em tempo de execução.

Com a programação orientada a aspectos, é possível inserir, de forma transparente, o serviço de replicação em um sistema computacional. Utilizando aspectos, o desenvolvedor pode se concentrar na resolução dos problemas pertinentes da aplicação que está criando, não precisando se preocupar com a tolerância à falhas. Esta é acrescentada ao sistema através de aspectos. O tempo de criação da aplicação é reduzido e a reusabilidade do código é garantida porque os módulos do sistema somente implementarão requisitos funcionais.

3.1 Inserção de Replicação Através de Aspectos

A figura 1 apresenta um diagrama de componentes que mostra a utilização de aspectos para inserir objetos específicos da aplicação (objetos críticos do sistema, que necessitam ser tolerantes a falhas) no cache de objetos. O mecanismo de replicação está implementado em um módulo chamado ReplicationService. Este implementa métodos que fazem uso da classe TreeCacheAop para estabelecer um cache em um ou mais nodos de um sistema distribuído. Se for utilizado somente um nodo, a tolerância a falhas não é alcançada porque este se torna um ponto único de falhas. Os objetos (POJO) inseridos neste cache são replicados em todos os nodos que formam o cache.

Para evitar inserir objetos da classe ReplicationService na aplicação, o que caracterizaria o espalhamento de código e a intrusão, aspectos são utilizados. Através de aspectos, pontos de junção da aplicação, especificados pelo desenvolvedor, são capturados e serviços da classe ReplicationService são executados. Estes serviços são a inserção, retirada e consulta de objetos no cache. A alteração do estado de um objeto que está presente no cache é realizada de forma transparente pela classe TreeCacheAop.

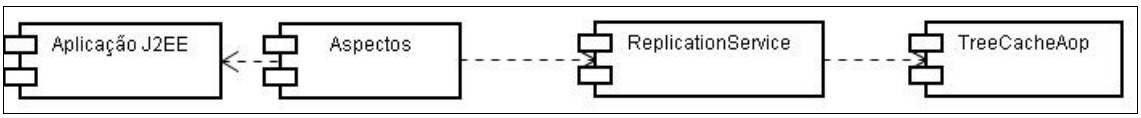

Figura 1 – Arquitetura para Utilização da POA

Este trabalho propõe um *framework* que disponibiliza aos desenvolvedores de

aplicações corporativas no ambiente web, o componente ReplicationService e modelos semi-prontos de aspectos (*templates*) concretos e abstratos. Aspectos abstratos definem funções genéricas e os aspectos concretos, que herdam as funcionalidades dos aspectos abstratos, podem ser definidos para situações particulares de uma aplicação. Assim, aspectos concretos podem ser criados para definir quais objetos críticos do sistema devem ser replicados. Os aspectos abstratos se responsabilizam por executar os serviços da classe ReplicationService. O desenvolvedor somente terá de realizar algumas alterações nos aspectos disponibilizados para poder fazer uso da replicação.

4. Conclusão

Este trabalho mostrou que um serviço de cache de objetos, configurado para ser replicado pode ser utilizado para prover tolerância a falhas. As API´s TreeCache e TreeCacheAop podem ser utilizadas em um servidor de aplicações J2EE (JBoss).

É também proposta a utilização de aspectos para inserir os serviços do cache de objetos em aplicações desenvolvidas na arquitetura J2EE. Este é um trabalho em andamento, que faz parte de um projeto maior cujo objetivo é desenvolver um *framework* que forneça diversos serviços de tolerância a falhas para aplicações desenvolvidas na arquitetura J2EE. Este *framework* disponibilizará aspectos prontos ou semi-prontos (*templates*) que simplificarão o uso de mecanismos de tolerância a falhas por desenvolvedores de aplicações web na plataforma J2EE.

Referências

Kurniawan, B. (2002) Java for the Web with Servlets, JSP and EJB: A Developer´s Guide to J2EE Solutions. New Riders Publishing.

Filman, R. et al. (2005) Aspect-Oriented Software Development. Addison-Wesley.

Penchikala, S. (2004) "Implementing Object Caching with AOP". http://www.theserverside.com/articles/article.tss?l=ObjectCachingWithAOP, September.

JBossCache Web Site, <http://www.jboss.com/products/jbosscache>

Ban, B. et al. "JBoss Cache TreeCache: A Structured, Replicated, Transational Cache". http://labs.jboss.org/portal/jbosscache/docs/1.3.0/TreeCache/en/html/index.html

JGroups Web Site, < http://www.jgroups.org>

Lippert M. Lopes, C. V. (2000). A Study on Exception Detection and Handling Using Aspect-Oriented Programming. In *Procedings of the 22nd International Conference on Software Engineering*, pages 418-427. ACM Press.

Kiczales, G. et al**.** (2001) Discussing Aspects of AOP, in *Communications of ACM*, v.44, n.10, p.33-38, October.

Kiczales, G. et al**.** (1997) Aspect-Oriented Programming, in *Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP '97)*, Finland, Springer-Verlag, 1997, pp. 220-242.