MAC 0434/5765 - Sistemas de Middleware Avançados
Segundo Semestre de 2004

Transações CORBA/XA para o JBoss

Neste trabalho você familiarizará com o lado interno de um gerenciador de transações simples[1], o mini-gerenciador de transações que faz parte do servidor de aplicações JBoss. Seus objetivos são:

- fazer com que o mini-gerenciador de transações coordene transações distribuídas envolvendo recursos CORBA e recursos XA,
- fazer com que componentes EJB implantados num servidor JBoss participem, como recursos CORBA, de transações coordenadas por um serviço de transações CORBA externo (que pode ser o gerenciador de transações de outro servidor JBoss),
- propagar o contexto transacional nas chamadas CORBA inter-servidores.

1 Gerenciamento de Transações no JBoss

O mini-gerenciador de transações do JBoss não tem *logging* (e portanto não faz recuperação de quedas do sistema) nem propaga o contexto transacional para outros servidores J2EE (ou seja, não trata transações distribuídas envolvendo vários servidores J2EE). Ele é voltado exclusivamente para transações XA, atuando como coordenador de transações distribuídas que envolvem somente recursos XA diretamente utilizados por componentes EJB implantados no servidor de aplicações. Bancos de dados e filas de mensagens são exemplos típicos de recursos XA.

Considere as três situações descritas abaixo.

1. Cenário 1: cada transação começa no servidor de aplicações, faz chamadas somente a EJBs implantados no próprio servidor e envolve apenas um gerenciador de banco de dados (que mantém o estado persistente de todos os *entity beans* usados pela transação).
2. Cenário 2: cada transação começa no servidor de aplicações, faz chamadas somente a EJBs implantados no próprio servidor e pode envolver mais de um gerenciador de bancos de dados ou de filas de mensagens.
3. Cenário 3: cada transação pode envolver EJBs implantados em servidores J2EE distintos, que se comunicam via IIOP.

O gerenciador de transações do JBoss é uma solução perfeitamente adequada para o cenário 1. A ausência de *logging* e recuperação de quedas no gerenciador de transações não causa problemas, pois cada transação envolve um único servidor de banco de dados (que tem seu próprio *log* e seu procedimento de recuperação de quedas do sistema).

No cenário 2 temos, para cada transação, um "two-phase commit" envolvendo apenas recursos XA. O gerenciador de transações do JBoss pode ser usado em tal cenário, mas não oferece proteção contra quedas do sistema. Ele não é adequado quando o cenário 2 aparecer numa aplicação tipo "missão crítica", a qual necessita de garantia absoluta de que mesmo em caso de queda do sistema, este não ficará num estado inconsistente, com alguma transação executada "pela metade". Não se deve usar um gerenciador de transações sem *logging* para uma aplicação de transferência de fundos entre contas bancárias em diferentes servidores de bancos de dados, por exemplo.

O gerenciador de transações do JBoss não oferece nenhum suporte para o cenário 3.

[1] A expressão "gerenciador de transações relativamente simples" seria mais adequada aqui, pois nenhum sistema desses (por mais simplificado que seja) é *realmente* simples.

1.1 O (Mini-)Gerenciador de Transações

O gerenciador de transações é implementado pelo MBean `org.jboss.tm.TransactionManagerService`, cujos fontes estão no subdiretório `transaction` da árvore de fontes do JBoss. Este MBean é uma "casca" bem fina ao redor da classe `org.jboss.tm.TxManager`, que por sua vez delega a maior parte de suas tarefas para a classe `org.jboss.tm.TransactionImpl`. É nesta classe que está o tratamento de "two-phase commit" envolvendo recursos XA.

1.2 Suporte (Restrito) para Transações CORBA

O MBean `org.jboss.tm.TransactionManagerService` é um embrião de serviço de transações CORBA. Os fontes desse MBean estão no subdiretório `iiop` da árvore de fontes do JBoss. Atualmente a única função desse MBean é dar suporte a demarcação de transações por parte de clientes RMI/IIOP (através da interface `javax.transaction.UserTransaction`) ou de clientes CORBA (através da interface IDL `Current`, definida no módulo `org.omg.CosTransactions`).

Note que o MBean `org.jboss.tm.TransactionManagerService` é uma "casca" bem fina ao redor da classe `org.jboss.tm.TransactionServiceImpl`, que implementa as interfaces do serviço de transações CORBA. Note, ainda, que o método `register_resource` da interface `Coordinator` lança a exceção `NO_IMPLEMENT`, com mensagem de detalhamento `"Two-phase commit is not supported"`.

2 O Que Você Deve Fazer

Seu objetivo é modificar o mini-gerenciador de transações do JBoss para que ele dê suporte ao cenário 3, sem oferecer proteção contra quedas do sistema. Suas tarefas incluem:

- Implementar o método `register_resource` da interface `Coordinator`. É através desse método que recursos CORBA se registrarão com o coordenador da transação.
- Fazer com que a classe `org.jboss.tm.TransactionImpl` conheça recursos CORBA (além dos recursos XA que ela já conhece) e trate o "two-phase commit" envolvendo os dois tipos de recursos.
- Assegurar a propagação do contexto transacional nas chamadas IIOP.
- Fazer com que o servidor de aplicações registre recursos CORBA com o coordenador de uma transação iniciada externamente. Chamadas a `register_resource` devem ocorrer quando EJBs forem chamados remotamente com contexto transacional, no primeiro uso de um recurso XA pelo método do EJB.

Dúvidas sobre este trabalho devem ser enviadas para a lista de discussão de SMA.

Bom trabalho!