Replicação de Servidores com Tolerância a Faltas *

José Martins, Ricardo Marques, João Leitão
TFD005 - Faculdade de Ciências da Universidade de Lisboa
{i26557, i26600, i28349}@alunos.di.fc.ul.pt

October 23, 2004

Abstract

A utilização de Replicação é uma das técnicas mais usadas hoje em dia não só para aumentar a disponibilidade de serviços, minimizando a latência em sistemas de grande dispersão geográfica mas também para garantir tolerância a faltas num sistema distribuido. O desenvolvimento de técnicas de comunicação em grupo facilita em certa medida a implementação de servidores replicados, minimizando a complexidade inerente a garantir consistência no estado das várias réplicas. Neste Artigo vamos sugerir uma possível concretização de um sistema simples de registos, baseado em servidores replicados que recorrem à comunicação em grupo, usando para isso a plataforma APPIA.

*Este trabalho foi realizado no contexto da cadeira de Tolerância a Faltas Distribuídas no ano Lectivo de 2004/2005 na FCUL, este trata-se de um artigo preliminar e por isso muitas questões não se encontram ainda analisadas com uma suficiente profundidade

1 Introdução

A Replicação [2] é uma técnica usada para garantir um aumento da escalabilidade/performance em vários sistemas distribuídos de grande escala, sendo um desses exemplo o das Bases de Dados [1], no entanto é também uma forma relativamente intuitiva de garantir que um sistema é tolerante à falha de um dos seus componentes, garantindo que todos os restantes podem continuar o seu normal funcionamento. Neste artigo analisamos o caso especifico de criar um servidor replicado, que guarda um conjunto de valores em registos. Este servidor, pela sua simplicidade, suporta apenas duas operações: *Ler* um valor de um registo, e *Escrever* um certo valor para um registo. Este sistema deve suportar dois modos de funcionamento, estes modos serão descritos em 2.1, qualquer destes modos deve no entanto, suportar a falha de réplicas, assim como ser capaz de reentigrar uma réplica. Este artigo encontra-se organizado da seguinte forma: na 2 descreve-se sumariamente a estrutura do sistema e os seus componentes. Em 3 analisa-se a pilha protocolar a usar na concretização do sis-

tema. 4 concluí este artigo.

2 Arquitectura

Possuímos um conjunto de réplicas de um mesmo servidor, que se conhecem mutuamente através da abstração de Grupo. Estas réplicas comunicam entre si através de troca de mensagens em canais não fiáveis, através de primitivas de multicast oferecidas pela comunicação em grupo. Existem um numero desconhecido de clientes, que contactam uma réplica para obter préstimos de serviços da mesma, através de troca de mensagens em canais não fiáveis. O Sistema será implementado sobre a plataforma Appia [4], um sistema de comunicação baseado em microprotocolos.

2.1 Servidor

Um servidor é um processo que possui localmente um conjunto de registos com valores inteiros. Quando um servidor é inicializado começa com um número pré determinado de registos; no caso de ser o primeiro servidor a arrancar os valores devem se encontrar todos inicilizados com 0 (zero), caso contrário deve efectuar um processo de join ao grupo de réplicas já activas e sincronizar os seus registos com essas mesmas réplicas, sendo que apenas após esta fase de arranque, pode começar a oferecer os seus serviços. O servidor suporta duas primitivas:

- *LER index*
- *ESCREVER index valor*

Um servidor tem dois modos de funcionamento, sendo que este é determinado no arranque da primeira réplica, todas as restantes devem ajustar o seu modo de funcionamento para o do funcionamento do grupo, ainda que no seu arranque tenham recebido instruções contrárias. Os dois modos possíveis são:

2.1.1 Propagação em melhor esforço

A diferença de tempo que pode existir entre a actualização das várias réplicas pode ser visível para os clientes. Isto é, um cliente pode ler um valor e de seguida, outro cliente, porque contacta outra réplica, ler ainda o valor anterior.

2.1.2 Propagação atómica

As actualizações devem aparentar ser atómicas. Ou seja, se um clinte vê uma actualização, qualquer cliente que leia esse registo no futuro deve ver essa actualização.

2.2 Comportamento do Servidor

Um servidor ao receber uma mensagem de um cliente, se esta for uma mensagem de leitura, simplesmente responde ao cliente com o valor requerido pelo mesmo. No caso de se tratar de um pedido de escrita e o funcionamento ser o de melhor esforço, este simplesmente redireciona para todo o grupo usando um broadcast fiavel uniforme da actualização requerida, qualquer servidor que receba do grupo um pedido destes, e caso se encontre em modo de Propagação melhor esforço, simplesmente efectua a actualização do registo, tão cedo quanto possível. No caso do modo de funcionamente de Propagação Atómica, a grande diferença de

comportamento reside no facto de que qualquer réplica ao receber um pedido de actualização de um registo por parte de um elemento do grupo de réplicas, termina todas as operações de leitura pendentes nesse momento, e após isto, e só após, processa o pedido de escrita, sendo que não processa qualquer outro pedido de leitura recebido após o pedido de escrita. Nos processos de escrita todas as réplicas memorizam o pedido de actualização mais alto que receberam de cada cliente, para poderem detectar pedidos duplicados. (que não redirecionam para o grupo, apenas confirmam o processamento ao cliente).

2.3 Comunicação

As réplicas encontram-se inseridas num Grupo, sabem por isso a cada momento quem são todas as outras réplicas existentes nesse grupo, através do serviço de Filiação do Grupo [2].

As réplicas comunicam entre si recorrendo a primitivas de comunicação em grupo, que de uma forma simples lhes permite enviar mensagens para os elementos do grupo através de Broadcast ou Multicast.

É usado o conceito de Sincronismo de Vista [2], em que uma vista representa um conjunto de processos num grupo, no caso do conjunto de elementos que pertencem ao grupo sofrer uma alteração (mudança de filiação do grupo) todos os elementos do grupo recebem uma nova vista, garantindo-se que todos os processos "Correctos" recebem o mesmo conjunto de mensagens dentro de cada vista isolada.

As réplicas comunicam entre si através de primitivas que garantem a ordem total de entrega de mensagens de forma uniforme, garantido que todos os processos do grupo recebem as mensagens pela mesma ordem.

2.4 Cliente

Um cliente é um processo interactivo, que recebe de um utilizador pedidos que podem ser para ler o valor de um registo, ou escrever um valor para um registo. Quando um cliente arranca, deve ser indicado a este uma réplica activa (e correcta) do servidor, este efectua pedidos ao servidor, recorrendo a pacotes UDP (comunicação não orientada à ligação), isto garante que a resposta ao pedido, pode ser recebida de uma réplica que não aquela a que o pedido foi originalmente efectuado, de uma forma transparente para o cliente. O cliente quando recebe uma mensagem de um servidor, recebe também em "piggy-back" um conjunto de outras réplicas activas, esta informação é guardada localmente, de forma a que no caso de não se receber uma resposta a um pedido ao fim de um periodo, esse pedido é repetido para uma outra réplica escolhida aleatoriamente. Os cliente apenas efectua um pedido de cada vez, sendo que o próximo pedido só é enviado após a recepção da resposta do pedido anterior. Os pedidos devem ser dotados de um identificador único que permita identificar a nível das réplicas pedidos de escrita duplicados.

3 Pilha Protocolar

Pretendemos efectuar a comunicação entre as várias réplicas recorrendo a uma composição de microprotocolos (já implementados para plataforma Appia) que irá possuir no seu topo a camada: TotalAbcastLayer,

seguida de UniformLayer, sob a qual se encontram todas as camadas que oferecem a abstração de Sincronismo na Vista, e no nível mais baixo o TCPCompleteLayer. A nossa sugestão de pilha protocolar baseia-se numa pilha utilizada para resolver o problema de Pong a 4 jogadores usando também a plataforma Appia como suporte para a comunicação [3].

4 Conclusões

Analisámos o problema de implementar um servidor replicado, que guarda um conjunto de valores em registos. Apresentámos o problema em 1, bem como os traços gerais da arquitectura sobre a qual nos debruçámos em 2. Descrevemos sumariamente dois modos de funcionamento em 2.1, bem como o comportamento que pretendemos implementar no servidor para cada um destes em 2.2. Descrevemos ainda, em traços gerais, a pilha protocolar que planeamos utilizar nesta concretização em 3.

References

[1] B. Kemme, G. Alonso, "A Suite of Database Replication Protocols based on Group Communication Primitives"

[2] P. Veríssimo, L. Rodrigues, " Distributed Systems for System Architects," Kluwer Academic Publishers, 2001

[3] J. Mocito, L. Rosa, N. Almeida, "Pong a quatro jogadores, distribuído e tolerante a faltas"

[4] L. Rodrigues, H. Miranda, A. Pinto, "Application program interface specification of *Appia*," version 1.2, Technical report, Universidade de Lisboa, July 2001