Servidor de Registos Replicado e Tolerante a Faltas

João Rebelo
i29078@alunos.di.fc.ul.pt

Hugo Gomes
i28739@alunos.di.fc.ul.pt

Luís Marques
i29242@alunos.di.fc.ul.pt

Abstract

A replicação de servidores é usada para obter tolerância a faltas. Este artigo descreve uma concretização da replicação de dados de um servidor com vista a optimizar para um uso típico com um número de leituras vastamente superior ao número de escritas.

1. Introdução

É importante para muitos serviços oferecer garantias de disponibilidade e consistência. Para tal, é necessário munir um sistema de redundância. Uma forma possível de redundância é a replicação de um servidor. A concretização da replicação tem implicações no desempenho obtido e pode beneficiar diferentes formas de uso.

Este artigo descreve uma concretização da replicação dos dados de um servidor com vista a optimizar um uso típico com um número de leituras vastamente superior ao número de escritas.

Este artigo tem uma estrutura bem definida. Inicialmente temos uma descrição detalhada do sistema. De seguida uma breve descrição da arquitectura. Na secção seguinte, tem uma explicação de como é que vai ser a comunicação entre os processos. A secção seguinte descreve os métodos de propagação a ser usados. Nas duas secções seguintes, temos a descrição mais detalhada dos dois diferentes métodos de propagação. As últimas secções tratam de descrever como é feita a eleição do coordenador, com é tratada tolerância a Faltas, como é feita a Detecção de Falhas, como é feito o Tratamento de Falhas e por ultimo como são inserido Novos elementos na vista. Na penúltima secção, contém uma descrição detalhada da pilha protocolar utilizada. Na última secção, apresentamos as nossas conclusões.

2. Descrição detalhada do sistema

Neste sistema iremos concretizar um servidor replicado de dados, sendo este constituído por uma lista de valores. Estes valores serão referenciados pelo seu índice, e terão um valor, que corresponderá à informação guardada.

Este servidor virtual será constituído por várias réplicas, sendo que entre elas existe uma que é responsável por dar a resposta ao cliente das operações de escritas. Tirando este facto não existe nenhuma distinção hierárquica, isto é, todas as réplicas comportam-se exactamente da mesma forma.

Iremos considerar que apenas serão possíveis dois tipos de operações, sendo elas a leitura do valor existente num determinado índice, e a escrita dum valor num determinado índice.

O sistema deverá funcionar em dois modos diferentes, sendo eles a ”Propagação melhor-esforço”, e a ”Propagação atómica”.

Na ”Propagação melhor-esforço” quando é realizado uma escrita numa réplica, esta será imediatamente visível para quem realizar uma leitura nessa réplica. Desta forma, um cliente que realize ao mesmo tempo uma leitura em dois servidores diferentes, poderá obter valores distintos. Obtém-se assim uma maior eficácia nas operações de escrita, podendo-se perder alguma consistência nas operações de leitura.

Na ”Propagação atómica” quando se realiza uma escrita, está será visível para os clientes como se tivesse ocorrido em cada réplica do servidor ao mesmo tempo. Desta forma qualquer leitura que seja feita em duas réplicas diferentes no mesmo instante deverá retornar o mesmo valor. Com isto obtém-se uma leitura mais consistente, sendo que dois clientes não poderão ler valores diferentes entre cada operação de escrita. Para obter esta propriedade é necessário comprometer o tempo de realizar a operação de escrita, fazendo com que esta apenas seja considerada acabada quando se tiver a certeza que todas as réplicas tem o valor correcto.

3. Arquitectura

A nossa proposta para este projecto consiste em desenvolver um sistema tolerante a faltas. Para este fim iremos usar os serviços fornecidos pela plataforma de comunicação Appia[**?**], baseados em micro protocolos, desenvolvido pelo departamento de informática da Faculdade de Ciências da Universidade de Lisboa.

O nosso sistema estará dividido numa arquitectura

cliente-servidor, sendo que apenas os servidores usarão os serviços prestados pelo Appia.

O cliente estará dividido em duas camadas, sendo elas a camada da aplicação (em que serão implementadas as várias operações que um cliente poderá fazer) e a camada de comunicação (que irá concretizar a comunicação com o servidor virtual). Esta comunicação entre o cliente e os servidores será realizada sobre o protocolo UDP / IP.

Cada processo servidor usa a plataforma Appia para se implementar a partir de várias camadas protocolares, sendo que grande parte delas são de carácter geral, e já se encontram desenvolvidas, enquanto que outras iremos desenvolver para o nosso problema específico.

4. Comunicação entre os processos

De forma a não sobrecarregar demasiado uma única réplica decidimos que o processo cliente irá escolher aleatoriamente um dos processos servidores que conhece. Desta forma acreditamos que os vários clientes se poderão distribuir equitativamente por cada servidor, não dando assim lugar a existir réplicas sobrecarregadas.

Como já foi referido atrás a comunicação realizada entre o cliente e o servidor será baseada no protocolo UDP. Este protocolo foi escolhido devido ao facto que cada pedido realizado pelo cliente será potencialmente realizado a réplicas diferentes. Desta forma o esforço existente de estabelecer uma ligação TCP não seria muito rentável.

Nos processos servidores a recepção dos pedidos de parte dos clientes será feita através dum canal do Appia, oferecendo umas determinadas garantias (QoS), sendo estas definidas posteriormente.

Como a comunicação entre os servidores será baseada no envio de mensagens em broadcast, então decidiu-se usar o protocolo UDP devido a facilidades fornecidas por este protocolo para este tipo de comunicação

5. Métodos de propagação

Nesta secção descreve-se as particularidades de cada modo de propagação, e como se pensa resolver os problemas inerentes a cada um.

5.1. Propagação melhor-esforço

Para a concretização deste modo iremos utilizar algumas das facilidades fornecidas pelos micro-protocolos já desenvolvidos para a plataforma Appia, bem como uma camada que será concretizada pelo grupo. As facilidades do Appia a ser usadas são a existência de comunicação fiável em grupo, e a correspondente noção de vista, e a ordenação total das mensagens nesta vista.

A camada (bestEffortLayer) que iremos implementar será responsável por receber os pedidos dos clientes (provenientes das camadas inferiores), entrega-los à aplicação, receber o resultado da aplicação, e devolve-lo ao cliente. Há que ter em atenção que no caso das operações de escrita apenas o coordenador do grupo irá responder. Ambos os pedidos são processados imediatamente, pois para uma leitura assume-se que todas as réplicas têm o valor que deverá ser respondido, e para o caso da escrita assume-se que todos os processos recebem o pedido de escrita.

5.2. Propagação atómica

Neste modo iremos usar as funcionalidades usadas para concretizar a ”propagação melhor-esforço” mais uma camada especifica criada por nos, para este modo de propagação.

Esta camada (atomicLayer) é responsável por manter a atomicidade das operações de escrita a serem realizadas. Com este fim, esta camada irá implementar um sistema de trancas. Sendo assim sempre que for recebido um pedido de escrita esta camada irá guardar todos os pedidos posteriores enquanto não tiver a total certeza que todas réplicas têm a nova alteração. Para que as réplicas determinem se as outras réplicas já receberam o pedido de escrita será enviado uma mensagem (Ack) broadcast para todas as outras réplicas. Quando uma réplica receber os Ack’s de todos os processos da vista, então o pedido de escrita será entregue à aplicação. A partir deste momento, os pedidos que haviam sido feitos irão começar a ser respondidos. As respostas dadas aos clientes serão enviadas, no caso dos pedidos de leitura, assim que se obter resposta da aplicação. No caso dos pedidos de escrita, o coordenador enviará, assim que receber o pedido, uma mensagem de confirmação de recepção, e logo que a escrita for efectuada (recepção de todos os Ack’s, e entrega da alteração à aplicação) o mesmo irá enviar uma mensagem de confirmação de pedido tratado.

6. Coordenador

Para determinar-mos o coordenador duma forma determinística baseamo-nos no conceito de vista. Sendo assim o algoritmo de eleição basear-se-á nas seguintes regras:

Se a réplica for a primeira da vista então assume-se como coordenador. Caso exista uma alteração na vista o primeiro processo da vista assume-se como coordenador. Caso exista uma alteração na vista o processo coordenador, deixará de o ser caso não seja o primeiro da mesma.

7. Tolerância a Faltas

Qualquer processo do conjunto de réplicas pode falhar. Alem disso pode haver também falhas na rede. Um con-

junto de processos pode assim, a qualquer momento, estar indisponível para participar na replicação do estado do servidor virtual.

7.1. Detecção de Falhas

É muito importante para o bom funcionamento deste protocolo que as várias réplicas sejam informadas das falhas que as outras réplicas tenham. Desta forma usamos o conceito de vista para determinar a existência de novos processos, bem como de processos que tenham abandonado o serviço.

7.2. Tratamento de Falhas

Caso se detecte a falha de uma réplica, durante um processo de escrita, teremos de ignorar o Ack que recebemos desta, ou verificar se estamos a espera de um Ack proveniente desta réplica, e ignora-la neste caso.

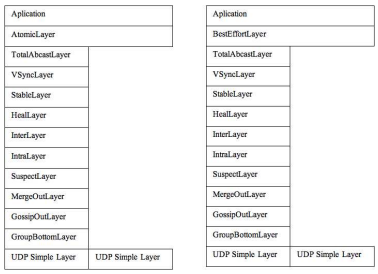
Existe um caso especial neste problema que será quando ocorre uma separação na rede. De forma a detectar estes casos, e trata-los convenientemente sempre que recebemos uma nova vista comparamo-la com a vista antiga, e caso o número de elementos da nova seja apenas uma minoria dos que existiam na vista antiga, então as réplicas deixarão de responder aos pedidos. Caso isto não se fizesse algumas das propriedades das propagações estariam comprometidas.

8. Novos elementos na vista

Quando uma réplica, ou várias se juntam a uma vista já existente, e em funcionamento, o coordenador estará responsável de enviar o seu estado para a vista. Apenas os novos processos irão entregar esta mensagem à aplicação. Caso após esta alteração da vista o coordenador falhar antes de enviar o estado, então o novo coordenador tratará de o fazer.

9. Pilha Protocolar

Como já foi referido acima neste trabalho iremos usar várias camadas que formarão a Qualidade de Serviço (QoS). Estes QoS irão variar conforme o modo de propagação, e conforme o tipo de pedidos a serem feitos. Sendo assim terá o seguinte aspecto:



Estas camadas garantem-nos a existência do conceito de vista para as aplicações, bem como da sincronia na mesma (da GroupBottomLayer à VsyncLayer). Garantem-nos também a existência de ordem total na entrega das mensagens, incluindo as de alteração de vistas (TotalAbcastLayer). As camadas superiores já foram descritas anteriormente.

10. Conclusões

Neste artigo descreveu-se uma forma de concretizar a replicação de um servidor replicado dando uso a uma plataforma já existente. Foi realizada uma descrição tão detalhada quanto possível da solução proposta, faltando agora comprovar na pratica a viabilidade destes algoritmos.

11. Trabalho Futuro

Neste artigo não foi comparado este sistema a outras variantes, tal como por exemplo sistemas em que não seria necessário escrever em todas as réplicas, mas que seria necessário ler em mais do que uma réplica.

References

[1] H. Miranda and L. Rodrigues. Application program interface specification of appia. Madeira Island, Portugal, 2001.