Servidor Replicado

Grupo 02: Ana Ferreira 29454, Hugo Vicente 30342, Pedro Costa 29685

October 23, 2004

Abstract: *Hoje em dia a necessidade é crescente de sistemas distribuídos de armazenamento de dados que sejam seguros e consistentes. A replicação dos dados é uma forma de construir um sistema tolerante a faltas. Faltas estas que adviriam da informação estar concentrada, que eventualmente se tornaria num ponto único de ruptura que comprometeria a disponibilidade do serviço. Este artigo irá apresentar um modelo de um sistema de dados replicado, que mantém a consistência, onde o cliente poderá realizar dois tipos de operações, leitura e escrita, de um modo transparente.*

1 Introdução

Desde sempre tem havido a necessidade de armazenamento de informação. Devido a isto começou-se por utilizar formas de armazenamento baseadas em arquivos físicos (ex.: arquivos em pastas e dossiers). Tal abordagem tornou-se inviável devido a requerer muita área e o seu manuseamento lento e difícil. Com o avanço da tecnologia, têm surgido alternativas digitais que tornam o processo de armazenamento e manipulação de dados mais simples e rápido. No início as organizações que necessitavam deste sistema, mantinham a informação toda concentrada numa só máquina, devido aos custos inerentes serem altos. Apesar desta solução aparentar ser vantajosa, tem inúmeros problemas. Exemplos podem ser o caso de a máquina ir a baixo, a informação deixa de estar acessível, ou poderá até perder-se. Além deste problema, as organizações começaram a expandir-se e como tal a necessidade de acederem a informação de qualquer sucursal aumentou, levando à investigação de novas abordagens. Com o aparecimento dos sistemas distribuídos [1][8], isto tornou-se possível. No entanto estes sistemas vieram trazer uma série de novos problemas. Nos sistemas distribuídos surgiram duas abordagens. Usando uma máquina que coordena todas as outras, ou distribuindo esta responsabilidade por todas as máquinas. A primeira solução apesar de ter um desempenho razoável, não resolve o problema de quando a máquina coordenadora vai a baixo comprometendo a informação. Já a segunda solução além de não ter este problema permite haver um balanceamento de carga por entre todas as máquinas. Na abordagem com a máquina coordenadora, esta fica responsável por manter a coerência de todas as outras, no entanto não havendo esse coordenador é necessário haver um consenso[1] entre elas. Uma forma de resolver este consenso vai ser apresentada neste artigo.

O artigo consiste em concretizar um servidor replicado. O servidor mantém apenas um conjunto de registos. Cada registo armazena um inteiro. O servidor pode funcionar em dois modos: Propagação Melhor Esforço(PME) e Propagação Atómica(PA)[8]. No primeiro modo 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. No segundo modo as actualizações aparentam ser atómicas, ou seja, se um cliente vê uma actualização, qualquer cliente que leia esse registo no futuro deve ver essa actualização. A concretização deste sistema será realizada com suporte no Appia[2][3][7]. O Appia oferece flexibilidade e modularidade que permite que as pilhas de comunicação sejam compostas e configuradas em tempo de execução, suportando também a coordenação de múltiplos canais[5].

O resto do artigo encontra-se organizado da seguinte maneira: na secção 2 será apresentada a arquitectura do sistema, na secção 3 encontram-se as componentes, nomeadamente os grupos, a ordenação, as faltas, a pilha protocolar, os eventos gerados, a estrutura do registo de dados, o modo de efectuar a propagação de melhor esforço e a propagação atómica. Na secção 4, o trabalho relaccionado, na secção 5, as conclusões e trabalho futuro e finalmente as referências.

2 Arquitectura

Estamos perante um sistema que visa uma interacção transparente entre cliente e servidor replicado de dados. Neste sistema decidimos implementar uma arquitectura em que não usamos coordenador. Desta forma distribuímos a responsabilidade por todos os servidores evitando que o coordenador seja um ponto de estrangulamento na comunicação e que o sistema esteja dependente da operacionalidade de uma só máquina. Cada servidor mantém um conjunto de registos constituídos por inteiros que inicialmente se encontram a 0. Sempre que uma alteração é efectuada numa das réplicas, esta terá de contactar as restantes a fim de lhes transmitir a operação para manter a consistência[8]. Para garantir este objectivo, as operações terão de ser realizadas pela mesma ordem em todos os servidores.

3 Componentes

3.1 Grupos

Todos os servidores que fazem parte do sistema replicado de dados pertencem a um grupo. Para a comunicação entre os elementos do grupo, serão utilizadas as camadas do Appia referentes a comunicação em grupo[3]. O Appia oferece ainda uma funcionalidade que podemos utilizar: a sincronia virtual[1][4]. Esta funcionalidade permite que todas as máquinas tenham noção dos constituintes do grupo em que estão inseridas através da entrega de vistas. Uma vista representa um conjunto de máquinas que fazem parte de um grupo. Sempre que existe uma alteração no grupo é entregue uma nova vista a todos os elementos.

Quando um novo servidor entra no grupo, todos os servidores recebem uma mensagem contendo uma nova vista actualizada. Comparando a vista passada com a actual, os servidores conseguem verificar que houve uma entrada para o grupo, e enviam os seus registos ao novo servidor. Este recebe os registos, actualiza-se e só depois começa a responder a pedidos.

Quando um servidor sai do grupo, os restantes servidores são notificados que esse elemento já não pertence ao grupo.

3.2 Ordenação

Nesta aplicação é imperativo que todas as réplicas apresentem os registos iguais. Para tal é necessário que todas as operações sejam efectuadas pela mesma ordem. Uma maneira

de atingir este objectivo é com ordenação das operações. Existem duas operações possíveis de ser utilizadas: leitura e escrita. No primeiro modo de propagação (PME) apenas as escritas necessitam de ordenação. A ordem para escritas vai ser a ordem total[1], pois este é um modo de propagar a actualização, ao mesmo tempo que garantimos que as actualizações são feitas em todas as máquinas pela mesma ordem. As leituras neste modo não irão ser ordenadas em relação as escritas pois não necessitamos de garantir a atomicidade. No segundo modo tanto as escritas como as leituras vão usar ordem total, porque necessitamos de ordenar as leituras em relação as escritas.

3.3 Faltas

Sendo esta uma aplicação distribuída é importante ter em atenção a possibilidade de ocorrência de faltas. Exemplos de faltas e respectivas atitudes a tomar são:

- O cliente tenta escrever um valor não inteiro (ex.: "xy"; "1.25"). Sempre que for recebida uma mensagem deste tipo, o servidor descartá-la-à.

- Sempre que uma máquina fique bloqueada à espera de uma mensagem mais do que um determinado tempo, dá-se um timeout e a ligação ou é quebrada ou é retransmitido o pedido.

3.4 Pilha Protocolar

Destas pilhas apenas as camadas ServBalanceLayer, CliBalanceLayer, ServApplLayer e CliApplLayer não são fornecidas pelo Appia.

As camadas ServBalanceLayer e CliBalanceLayer são responsáveis pelo mecanismo de balanceamento de carga no servidor e no cliente respectivamente. O modo de funcionamento encontra-se descrito mais a baixo neste artigo (Secção 3.6 e 3.7). A camada CliApplLayer é responsável pela interacção com o cliente, enquanto que a camada ServApplLayer vai tratar os pedidos feitos pelo cliente.

(servidor)

<table>
<tr><td colspan="2">ServApplLayer</td></tr>
<tr><td colspan="2">TotalAbcastLayer</td></tr>
<tr><td colspan="2">VSyncLayer</td></tr>
<tr><td colspan="2">LeaveLayer</td></tr>
<tr><td colspan="2">StableLayer</td></tr>
<tr><td colspan="2">HealLayer</td></tr>
<tr><td colspan="2">InterLayer</td></tr>
<tr><td colspan="2">IntraLayer</td></tr>
<tr><td colspan="2">SuspectLayer</td></tr>
<tr><td colspan="2">MergeOutLayer</td></tr>
<tr><td colspan="2">GossipOutLayer</td></tr>
<tr><td colspan="2">GroupBottomLayer</td></tr>
<tr><td>ServBalanceLayer</td><td rowspan="2">TCPCompleteLayer</td></tr>
<tr><td>UDPSimpleLayer</td></tr>
</table>

(cliente)

<table>
<tr><td colspan="2">CliApplLayer</td></tr>
<tr><td>CliBalanceLayer</td><td rowspan="2">TCPCompleteLayer</td></tr>
<tr><td>UDPSimpleLayer</td></tr>
</table>

3.4.1 Eventos

Para além dos eventos[7] definidos pelo Appia vamos ainda implementar os seguintes eventos:

- *Leitura:*

  O evento é gerado quando a camada de aplicação do cliente faz um pedido de leitura. O servidor ao responder gera um novo evento de resposta à leitura requerida pelo cliente.

- *Escrita:*

  O evento é gerado quando a camada de aplicação do cliente faz um pedido de escrita. O servidor gera um evento de recepção, confirmação ou de erro na escrita (caso tenha ocorrido uma falta).

- *Gestão de Carga:*

  O evento é gerado quando o cliente pretende realizar o pedido de estabelecimento de uma ligação. O servidor ao receber este evento cria um novo evento respondendo ao cliente com o número de clientes ligados.

- *Gestão de Coerência:*

  Quando o servidor recebe uma vista com um novo servidor, é gerado um evento que tem como objectivo actualizar o novo servidor.

3.4.2 Canais

Nestas pilhas vamos ter os seguintes canais[7]:
Melhor Esforço

- Na comunicação servidor-servidor utilizamos as seguintes camadas:

  - ServApplLayer
  - TotalAbcastLayer
  - Sincronia na Vista
  - Comunicação em Grupo
  - TCPCompleteLayer

- Na comunicação servidor-cliente:

  - Para um evento de leitura

Servidor:	**Cliente:**
ServApplLayer	CliApplLayer
UDPSimpleLayer	UDPSimpleLayer

  - Para um evento de escrita

Servidor:	**Cliente:**
ServApplLayer	CliApplLayer
TCPCompleteLayer	TCPCompleteLayer

  - Para um pedido de balanceamento de carga

Servidor:	**Cliente:**
ServApplLayer	CliApplLayer
ServBalanceLayer	CliBalanceLayer
UDPSimpleLayer	UDPSimpleLayer

Propagação Atómica

- Na comunicação entre servidores:

  - ServApplLayer
  - TotalAbcastLayer
  - Sincronia na Vista
  - Comunicação em Grupo
  - TCPCompleteLayer

- Na comunicação servidor-cliente:

  - No caso de leitura e no caso de escrita:

Servidor:	**Cliente:**
ServApplLayer	CliApplLayer
ServBalanceLayer	CliBalanceLayer
UDPSimpleLayer	UDPSimpleLayer

  - Para um pedido de balanceamento de carga:

Servidor:	**Cliente:**
ServApplLayer	CliApplLayer
TCPCompleteLayer	TCPCompleteLayer

3.5 Estrutura do registo de dados

Os dados são do tipo inteiro e estão organizados numa estrutura. Esta estrutura para cada índex terá o valor do inteiro e a respectiva versão. A versão será mais útil na propagação

melhor esforço como será descrito na secção seguinte.

3.6 Propagação Melhor Esforço (PME)

Neste ponto vamos descrever o modo de acção do servidor quando se encontra em PME.

No caso de uma leitura:

O cliente envia uma mensagem em multicast para o grupo de servidores com o pedido de leitura. Os pedidos de leitura são todos transmitidos por UDP. Estes respondem com o valor do pedido e respectiva versão numa mensagem UDP. Se esgotado um timeout e o cliente não ter recebido nenhuma resposta, volta a realizar o pedido. Caso o cliente receba versões diferentes escolhe a resposta com a maior versão.

Neste caso a realização de uma leitura não necessita de ser processada pela camada de ordem total, ao contrário das escritas. Como tal, vão ter um canal diferente, logo nunca são bloqueadas, podendo as leituras serem realizadas mais rapidamente que as escritas.

No caso de escrita:

A camada responsável pelo balanceamento de carga do cliente (CliBalanceLayer) envia uma mensagem em multicast para o grupo de servidores com um pedido de ligação. Nos servidores a camada ServBalanceLayer responde com o número de ligações que tem. Os pedidos de balanceamento de carga e respectivas respostas são efectuados por UDP. No cliente a CliBalanceLayer vai escolher o servidor com menos ligações activas e estabelecer uma ligação TCP a este. Com a ligação concretizada é enviado o pedido de escrita. O pedido ao chegar ao servidor é ordenado segundo a ordem total, deste modo assegurando que todos os servidores executam os pedidos pela mesma ordem. Aquando da concretização da escrita de um valor, um número de versão sequencial é associado ao registo que será consistente para todo o grupo.

3.7 Propagação Atómica (PA)

O mecanismo de execução tanto das leituras como nas escritas da PA será igual ao das escritas na PME. Neste modo as escritas e as leituras são transmitidas por TCP.

Para assegurar a atomicidade das operações alteramos a forma como são executadas as leituras em relação à propagação melhor esforço. Obrigamos então a que as leituras sejam ordenadas com as escritas, devido à necessidade de a leitura ter sempre que recair sobre a última escrita, fazendo assim com que o sistema se comporte de uma forma atómica.

4 Trabalho relacionado

A comunicação em grupo através da recepção de vistas foi estudada por José Pereira, Luís Rodrigues e Rui Oliveira[4]. Existem várias versões de implementação da ordem total, feitas algumas por alunos[2] e outras por professores[6].

5 Conclusões e Trabalho Futuro

Os sistemas de armazenamento de dados replicados são utilizados em várias aplicações hoje em dia. Têm por características principais a manutenção por parte de todas as réplicas de um único estado. Ou seja, mantêm a consistência. Neste artigo apresentámos um sistema que pode operar em dois modos: Propagação Melhor Esforço (PME) e Propagação Atómica (PA). Estes dois modos

são semelhantes, apresentando apenas algumas diferenças.

Na PME as operações de escrita não estão ordenadas com as de leitura. As escritas estão entre si ordenadas com ordem total e as de leitura, também entre si, com ordem FIFO[8]. Isto vai levar a optimizar as operações de leitura permitindo realizar estas operações mais rapidamente que as de escrita. Tal é possível pois as alterações feitas neste modo não necessitam de aparentar ser imediatas. Para optimização de leituras, estas são realizadas enviando uma mensagem de broadcast para o grupo. As escritas são feitas por TCP.

Na PA as duas operações estão ordenadas por ordem total. Deste modo não existe a possibilidade de uma operação de leitura ser realizada mais rapidamente do que uma escrita, fazendo com que as leituras retornem sempre o valor da última escrita. Neste modo as duas operações serão realizadas por TCP.

Apesar de não ser referido anteriormente no artigo, seria útil a implementação de log's. Ao haver uma actualização cada servidor guardaria os seus registos para no caso de todos os servidores irem a baixo, o primeiro a tornar-se operacional poderia usar este registo em disco para se actualizar, já que não existiriam outros servidores que lhe enviassem os registos, como aconteceria numa situação normal.

References

[1] Paulo Veríssimo e Luís Rodrigues, "Distributed Systems for System Architects", Kluwer Academic Publishers ISBN 0-7923-7266-2

[2] URL: http://appia.di.fc.ul.pt

[3] Pinto, "Appia Group Communication Manual", 2001.

[4] José Pereira, Luís Rodrigues e Rui Oliveira, "Reducing the Cost of Group Comunication with Semantic View Synchrony"

[5] H. Miranda, A. Pinto, and L. Rodrigues. "Appia, a flexible protocol kernel supporting multiple coordinated channels. In Proceedings of the 21st International Conference on Distributed Computing Systems"

[6] "An Indulgent Uniform Total Order Algorithm with Optimistic Delivery" Pedro Vicente e Luís Rodrigues

[7] "Application Program Interface Specification of Appia version 1.2" Hugo Miranda, Alexandre Pinto, Luís Rodrigues

[8] "Distributed Systems: Principles and Paradigms", Andrew S. Tanenbaum, Maarten van Steen