Replicação de servidores: uso de propagação atómica e melhor-esforço

André Rijo
Departamento de Informática
i28017@alunos.di.fc.ul.pt

José Antunes
Departamento de Informática
i29166@alunos.di.fc.ul.pt

Mário Henriques
Departamento de Informática
i29131@alunos.di.fc.ul.pt

Abstract

Os sistemas distribuídos estão presentes no dia a dia da maior parte dos cidadãos, e são responsáveis por operações muito delicadas. Devido à natureza destas operações, foi necessário a certa altura na história dos sistemas distribuídos, dotá-los com a capacidade de tolerar faltas.

Neste artigo é apresentada uma possível solução para replicação de dados entre servidores, permitindo que no caso de um servidor falhar os seus dados nunca serão perdidos, pois estes encontram-se replicados noutros servidores.

1 Introdução

Uma das grandes faltas nos sistemas distribuídos é quando um servidor, por uma qualquer razão, deixa de responder. Quando isto sucede, e se o servidor não recuperar em tempo útil, poderá provocar graves danos no sistema onde se encontra inserido. Por exemplo, se um servidor que contem dados importantes para o funcionamento de todo o sistema falhar, todos os dados neles contidos poderão ser perdidos, provocando a paragem de todo o sistema.

Neste artigo vamos apresentar uma solução que permita, em particular, resolver este tipo de problemas. Esta solução baseia-se na replicação de dados entre os servidores. Desta forma, se um servidor que contem dados importantes falhar, estes não irão ser perdidos. O sistema poderá recuperar desta falha acedendo a outro servidor do grupo, onde os dados anteriormente perdidos estão replicados. Assim, o sistema continuará o seu funcionamento, ocultando a falha ocorrida a qualquer elemento exterior.

Para a concretização da solução apresentada, vamos utilizar um sistema de vários servidores, onde existe noção de grupo fornecida pelo Appia [1] [2] [3]. Cada servidor possui localmente um conjunto de registos. Neste sistema é permitida a leitura e escrita de dados nos servidores por parte de clientes, e sempre que um destes escreva um valor no conjunto de registos de um determinado servidor, este valor irá ser replicado pelos restantes elementos do grupo. A leitura de um determinado registo num servidor é trivial, isto é, quando um servidor é confrontado com um pedido de leitura efectuado por um cliente, este simplesmente envia-lhe como resposta o valor contido no registo pretendido. A arquitectura do sistema irá ser descrita, com maior detalhe, na secção 2.

Na secção 3 é descrita a pilha protocolar e os respectivos algoritmos a implementar.

2 Descrição da arquitectura

O sistema estará estruturado de acordo com a Figura 1.

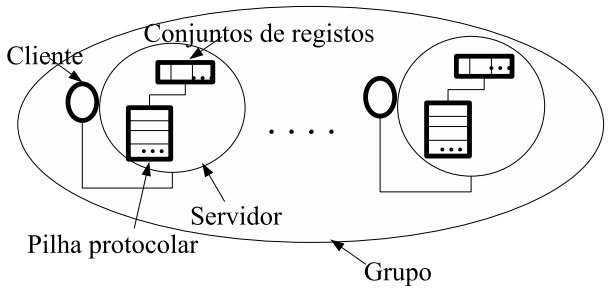

Figure 1. Esquema da arquitectura do sistema

Cada servidor apresenta um conjunto de regis-

tos, onde vão ser guardados os valores escritos pelos clientes. Este conjunto de registos não irá ser mais do que um vector de inteiros. Os registos de um servidor poderão ser consultados e alterados por um cliente. Para facilitar a interacção entre clientes e servidores, um cliente e um servidor serão agrupados no mesmo processo. Assim, sempre que um cliente quer consultar os dados chama a camada aplicação do processo e esta retorna-lhe os valores pretendidos. Contudo, no restante artigo vamos referir-nos ao servidor e ao cliente como se fossem dois processos independentes.

Todos os servidores estão contidos no mesmo grupo. Cada um deles, com ajuda do Gossip [1] [2] [3], consegue obter várias informações sobre o grupo, entre elas, todos os elementos que entram e saem do grupo.

3 Descrição da pilha protocolar e algoritmos

A pilha protocolar que vamos implementar esta representada na Figura 2.

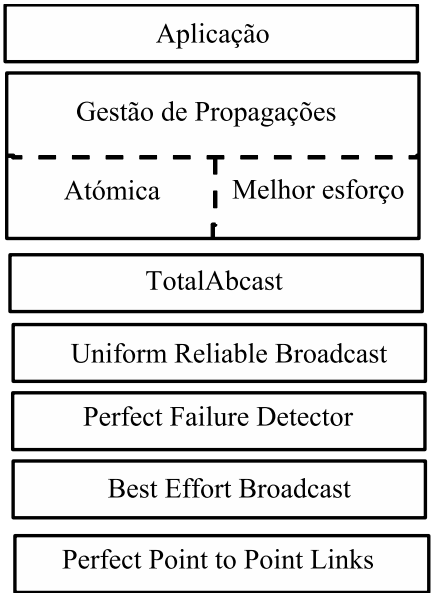

Figure 2. Pilha Protocolar

Nas subsecções 3.1 e 3.2 é feita uma descrição sucinta das camadas que compõem a nossa pilha protocolar.

Na subsecção 3.3 são descritos os algoritmos necessários ao funcionamento do sistema.

3.1 Camada Aplicação

Esta camada é a responsável pela resposta aos pedidos dos clientes, pedidos estes que podem ser de leitura e escrita. Esta possui um conjunto de registos, onde serão guardados inteiros, que poderão ser lidos e modificados pelos clientes. Um pedido de escrita é representado por dois valores: o primeiro (X) refere-se ao índex no conjunto de registos, o segundo (Y) refere-se ao valor a ser escrito na posição X. Assim o pedido vai ter o seguinte formato: ”ESCREVER X Y”. Por sua vez, um pedido de leitura é acompanhado de um único valor (X), representando o índex no conjunto de registos a ser lido, logo o formato da leitura é: ”LER X”.

Quando é feito um pedido de leitura do índex X (”LER X”) por parte de um cliente, a camada aplicação acede ao registo com o índex X, devolvendo ao cliente o valor contido nesse registo.

Ao receber pedido de escrita do cliente (”ESCREVER X Y”), esta camada apenas se preocupa em desencadear um conjunto de acções, enviando um evento para a camada abaixo (Gestão de propagações), com o intuito de actualizar o conjunto de registos de todos os servidores.

Com a recepção de um evento escrita (que tem a mesma informação do pedido de escrita feito pelo cliente) lançado pela camada ”Gestão de propagações”, a aplicação vai escrever no registo de índex X, o valor Y.

3.2 Camada Gestão de Propagações

Genericamente, esta camada é a responsável pela gestão da propagação da informação pelas réplicas. Esta pode implementar dois tipos de propagação, dependendo da opção ao se iniciar um servidor. Todos os servidores são obrigados a ter o mesmo modo de propagação, o que vai ser garantido pelo algoritmo de INSERÇÃO/REINSERÇÃO (descrito na secção 3.3.3).

Seguidamente, vão ser descritas com maior detalhe as duas camadas correspondentes a cada tipo de propagação.

3.2.1 Camada Propagação Melhor-Esforço

Esta camada ao receber um evento vindo da camada aplicação a indicar um pedido de escrita, é responsável pela passagem do evento para as camadas abaixo para que este seja enviado para as restantes réplicas do grupo, incluindo o emissor.

No caso de receber um evento vindo das camadas abaixo, e caso este evento seja um pedido de escrita, este será enviado para a camada aplicação, para que esta escreva no seu conjunto de registos, na posição pretendida, o valor indicado.

Esta camada vai respeitar o algoritmo PROPAGAÇÃO MELHOR-ESFORÇO descrito na secção 3.3.1.

3.2.2 Camada Propagação Atómica

Esta camada ao receber um evento vindo da camada aplicação a indicar um pedido de escrita, passa o evento para as camadas abaixo para que este seja enviado para as restantes réplicas do grupo, incluindo o emissor.

Quando é recebido um evento nesta camada vindo das camadas abaixo, esta vai guardar em buffer o valor a ser escrito, que será entregue posteriormente à aplicação.

Esta camada vai respeitar o algoritmo PROPAGAÇÃO ATÓMICA descrito na secção 3.3.2.

3.3 Descrição dos algoritmos

3.3.1 Algoritmo da Propagação Melhor-Esforço

A Propagação melhor-esforço consiste em difundir os dados recebidos num servidor pelas suas réplicas. Neste tipo de propagação, 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, contacta outra réplica, e ler ainda o valor anterior.

Neste projecto vamos tentar implementar este tipo de propagação respeitando a ordem com que os processos recebem as escritas dos clientes. Para tal vamos recorrer a implementação de uma camada do Appia que implementa a ordem total [4], que garante que todos os processos recebem as mensagens pela mesma ordem.

Este protocolo funciona da seguinte forma: como não é obrigatório que todos os clientes vejam o mesmo valor nas diferentes réplicas no mesmo instante, a alteração dos valores apenas tem que respeitar a ordem total das mesmas.

Quando um servidor tem uma alteração para efectuar, a camada aplicação pede à camada responsável pela propagação melhor-esforço que trate de difundir a mesma pelas restantes réplicas. Esta mensagem é enviada para todos os elementos do grupo incluindo o emissor. Desta forma e sabendo que existe ordem total, fornecida por uma camada do appia TotalAbcast [5], as alterações são efectuadas por todos os elementos do grupo pela mesma ordem, independentemente do tempo que as mensagens demoram a serem entregues. Desta forma, se dois servidores enviarem uma mensagem m1 e m2 e se um processo entregar m2 e m1 então todos os processos vão entregar m2 e m1. Assim podemos assegurar que todas as alterações vão ser realizadas pela mesma ordem em todas as réplicas, como se pode ver na Figura 3.

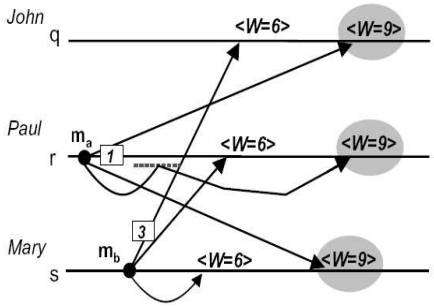

Figure 3. Ordem Total

3.3.2 Algoritmo da Propagação Atómica

A propagação atómica consiste em difundir os dados recebidos num servidor pelas suas réplicas. Neste tipo de propagação se um cliente vê uma actualização, qualquer cliente que leia esse registo, no futuro, deve ver essa actualização.

Quando um servidor tem um valor para escrever num dado registo, o cliente que fez esse pedido irá ficar

bloqueado até que todas as réplicas do grupo escrevam esse novo valor no seu conjunto de registos. O cliente vai ser desbloqueado quando receber uma mensagem de confirmação do servidor a indicar o sucesso ou insucesso da escrita.

Sempre que um servidor recebe um pedido de um cliente envia um evento para as camadas abaixo, para este ser enviado para os restantes elementos do grupo, incluindo o emissor. Se esta camada recebe um evento de escrita, da camada abaixo, guarda este evento num buffer até que uma nova vista[1] seja recebida. Sabendo que este evento é entregue por ordem total, não é necessário proceder à ordenação dos eventos. Os eventos que se encontram em buffer vão ser entregues à camada aplicação no momento em que é recebida uma nova vista, porque neste momento todas as réplicas do grupo encontram-se sincronizadas, isto é, todas as réplicas recebem o evento "ao mesmo tempo". Desta forma o valor é escrito em todas as réplicas de forma atómica.

3.3.3 Algoritmo Inserção/Reinserção de servidores

Este algoritmo tem como função a integração de um servidor no grupo, sendo responsável pela actualização do conjunto de registos do novo elemento.

Para que um novo elemento possa entrar no grupo, é necessário verificar se o seu modo de funcionamento, ou seja, se o tipo de propagação é o mesmo do resto do grupo. Caso seja o primeiro a entrar no grupo, o seu modo de funcionamento será o estabelecido para o grupo; caso já existam elementos no grupo e o seu modo de funcionamento seja diferente destes, é lançada um aviso a indicar que o modo de funcionamento não é compatível com o do grupo.

Se o novo servidor for admitido no grupo, vai proceder-se à actualização do seu conjunto de registos. Para tal é eleito um membro do grupo para partilhar o seu conjunto de registos com o novo elemento.

Essa eleição é realizada recorrendo ao par Ip:Porto de todos os elementos do grupo, isto é, dado um par x.y.z.w:p, quem tiver o menor número zwp é o eleito para actualizar o novo membro. Esta actualização é realizada recorrendo à sincronia na vista. Quando é recebida a primeira vista após a entrada do novo elemento, o servidor eleito aguarda a próxima vista para proceder à actualização do novo elemento. Durante o processo de actualização toda a comunicação servidor-cliente é suspensa até à chegada de uma nova vista.

[1] Conjunto dos pares IP:PORTO de todos os servidores.

4 Conclusão

Neste artigo foi apresentada uma solução para a difusão de dados entre os servidores de um grupo, através de dois tipos de propagação, a propagação atómica e a propagação de melhor esforço. Com esta solução, mesmo que um servidor falhe, os seus dados nunca irão ser perdidos, pois encontram-se replicados noutro servidor do grupo.

Foi também apresentada uma solução de integração/reintegração de novos servidores no grupo. Esta solução permite que todos os servidores que entrem ou reentrem no grupo fiquem com o conjunto de registos actualizado com o das restantes réplicas do grupo.

As soluções apresentadas permitem que se no caso de um servidor do grupo falhar, o sistema não irá ser afectado com a perda de dados.

References

[1] N. Carvalho, L. Rodrigues, *Implementing Reliable Broadcast Protocols in Appia*, 3rd November 2003

[2] Alexandre Pinto, *Appia Group Comunication Manual*, February 2001

[3] H.Miranda, A. Pinto, L. Rodrigues, *Application Program Interface Specification of Appia*, July 2001

[4] P. Verissímo, L. Rodrigues, *Distributed Systems for System Architects, Kluwer*, January 2001

[5] B. Simões, F. Vicente, P. Sousa, *TotalAbcast*, appia.di.fc.ul.pt/dist/index.html, 1st March 2001