TAP - Tolerance Advance Publisher-Subscriber
Grupo 08

André Nogueira 29505 | Paulo Levi 30817 | Simão Fontes 30820

30 de Outubro de 2006

Resumo

Com a evolução das aplicações informáticas a utilização de técnicas de tolerância a faltas, em sistemas distribuídos, têm se tornado vitais. Foi proposto na cadeira de tolerância a faltas distribuídas a criação de um sistema de Edição-Subscrição que assegura a execução do mesmo, independentemente da ocorrência de faltas.

1 Introdução

O trabalho proposto consite no desenvolvimento de um sistema edição-subscrição, em que a base é a existência de tópicos que agrupam informação. Nestes sistemas consideramos dois tipos de clientes:

- O Editor
- O Subscritor

A comunicação é efectuada de forma assíncrona por troca de mensagens. No caso dos editores estes têm a função de publicar informação sobre um determinado tópico. A informação publicada vai ser recebida por outro tipo de utilizador, o subscritor. Este vai receber apenas a informação dos tópicos que subscreve. Originalmente, o sistema Edição-Subscrição apresenta a existência de um mediador, onde os editores publicam a informação e os subscritores a subscrevem. No caso deste projecto e de modo a tornar o sistema mais simples vamos eliminar a existência do mediador e comunicar de forma directa e fiável. Tendo em conta que a base deste sistema é relativamente simples e fácil de implementar, vamos aumentar a complexidade, implementando técnicas de tolerância a faltas em sistemas distribuidos, que são aplicáveis em sistemas mais complexos.
O resto deste artigo apresenta-se organizado da seguinte forma:
Na secção 2 apresenta-se o problema e a ferramenta a utilizar para o resolver, na secção 3 a relação entre clientes e servidores, a secção 4 refere-se a conceitos importantes para a resolução do problema, a secção 5 , a faltas e ao modelo de faltas relativo a esta aplicação, na 6 apresenta-se a arquitectura utilizada para resolver o problema , na secção 7 consideramos que trabalho a realizar no futuro, e finalmente na 8, conclui-se o artigo.

2 Descrição do problema

O sistema a implementar não tem limite máximo de clientes conectados, existe sim, um parâmetro de configuração que limita o número máximo de grupos que o servidor pode criar. O projecto seguirá o modelo de comunicação em grupo.

2.1 Appia

No projecto em desenvolvimento vamos utilizar uma *framework* chamada Appia[3]. Esta não é mais que uma pilha de protocolos de comunicação por camadas, que são independentes entre si. Cada camada descreve o comportamento de um protocolo através de eventos. Estes eventos são usados para a comunicação entre o canal e as sessões. Os canais consistem numa instanciação de um conjunto ordenado de protocolos. Uma sessão é uma instância de um protocolo. O Appia já implementa vários modelos de computação distribuída, que são utilizados pelo nosso grupo no desenvolvimento deste sistema.

3 Comunicação entre intervenientes

3.1 Cliente - Servidor

Para um cliente poder integrar um grupo de um dado assunto terá que começar por comunicar com outro grupo, um grupo de servidores, o qual têm a responsabilidade de lhe devolver o nome do grupo requisitado. Toda a comunicação é feita sobre o modelo de *multicast* e não ponto a ponto.

3.2 Cliente - Cliente

A interacção entre clientes só é feita após a obtenção de um nome de um grupo. Após isto, o cliente agrega-se ao grupo e passa a ser diferenciado no tipo de cliente que é:

- Subscritor - ficar à espera de mensagens enviadas dos editores.
- Editor - vai publicar informação por meio de mensagens.

A comunicação é enviada pelo editor para o grupo seguindo o modelo *multicast*.

3.3 Servidor-Servidor

A interacção entre servidores serve para garantir a replicação da informação. Informação esta que contém os grupos e assuntos que nestes existem. A necessidade de replicação está em garantir a continuidade do serviço.

4 Conceitos

4.1 Noção de Grupo

Um grupo[1] consiste numa agregação de vários utilizadores que comunicam entre si. Os grupos suportam várias operações como a entrada/saída de elementos e obtenção dos elementos que pertencem ao grupo num dado instante. Os elementos de um grupo num dado instante é designado por vista de grupo. A comunicação dentro do grupo pode ser feita ponto a ponto ou por *multicast*. Esta funcionalidade é disponibilizada pela *framework* Appia e será utilizada em várias interacções.

4.2 Sincronia Virtual

Este mecanismo garante broadcast fiável, ou seja, que uma mensagem m quando enviada para uma vista de um grupo, é recebida por todos os processos correctos da mesma[2]. Garante também que antes de instalar uma nova vista todas as mensagens pertencentes a vista actual serão entregues aos processos. Este mecanismo também é disponibilizado pelo Appia.

4.3 Ordem Total

Consiste numa forma de garantir a entrega de mensagens[1]. Isto é, quando um processo pretende enviar várias mensagens, todos os processos que recebam estas mensagens as entregam todas pela mesma ordem. Existem outros modelos de ordenação mas tendo em conta o enunciado proposto não será utilizado qualquer outro.

4.4 Modelo de replicação

A forma de garantir a tolerância a faltas por parte dos servidores, é replicando o estado do sistema pelos mesmos. O método que optamos foi a replicação activa[1], que consiste em o cliente enviar uma mensagem para todas as réplicas, estas alteram o seu estado e respondem. Cabe ao cliente assimilar as respostas e estados do sistema.

5 Faltas

Em qualquer sistema podem ocorrer estados de erro que são provocados por faltas. Em sistemas distribuídos as faltas mais importantes dividem-se em dois grupos: externas e internas. As externas consistem em faltas que afectam principalmente a comunicação, faltas internas são aquelas que afectam os programas em execução[1].

5.1 Modelos de Faltas

Considerando o projecto e a disciplina em que este se enquadra vamos considerar apenas alguns tipos de faltas. Em particular vamos abordar faltas por omissão e falha de servidores, implementado o sistema de forma a tornar as faltas transparentes para o utilizador.

6 Arquitectura

A arquitectura que vamos tentar implementar neste projecto baseia-se nos seguintes pressupostos:

- Existe um grupo de servidores que presta serviços aos utilizadores. Este grupo ao ser baseado no protocolo de comunicação do Appia vai automaticamente atribuir a função de coordenador a um dos elementos do grupo.
- Todos os clientes ligam-se ao grupo de servidores para obter informações, esta agregação vai ser efectuada individualmente. O cliente abandona o grupo depois de obter a informação pretendida.
- Mediante o tipo de cliente vamos ter formas diferentes de interagir com os servidores:
  - Se for um subscritor apenas pede o nome do grupo de um determinado assunto.
  - Se for um editor este envia aos servidores o assunto do grupo e a qualidade de serviço que pretende, o coordenador devolve o nome do grupo ao qual o editor se deve associar.
- Caso ocorra uma falha no coordenador, o grupo de servidores vai eleger um novo coordenador, se isto não for possível, será impossível continuar a prestar serviços.
- Caso um subscritor entre no grupo de servidores, e o assunto que pediu não exista o grupo de servidores responde que o grupo ainda não foi criado.
- Caso o editor de um assunto sair do grupo em que se encontra, não vamos eliminar os subscritores desse assunto do grupo.
- A utilização da comunicação *multicast* garante que todas as réplicas vão receber as mesmas mensagens, e assim vão manter o mesmo estado.

O estado das réplicas apenas é replicado quando queremos juntar um novo servidor no grupo já existente. Neste caso o sistema não permite a ligação de mais clientes enquanto as réplicas não acabam a actualização. A comunicação entre clientes é efectuada da seguinte forma: existe um editor que depois de saber o nome do grupo ao qual se vai ligar, agrega-se a este grupo. Caso o grupo ainda não exista vai criar um grupo com o nome dado. Os subscritores agregam-se ao grupo e esperam comunicação do editor. Existe a possibilidade de num grupo existir mais que um assunto, podendo assim haver mais que um editor. Neste caso os subscritores vão ter de filtrar as mensagens que recebem. Adoptamos uma arquitectura bastante simples para colmatar o problema, de modo a simplificar a implementação, não dando ênfase a problemas mais complexos.

6.1 Pilha Protocolar

Considerado uma implementação sobre a *framework* Appia vamos tirar partido da pilha protocolar oferecida por esta. A pilha oferecida pelo Appia já oferece: comunicação em grupo com ordem total e sincronia virtual. No caso do servidor propomos a seguinte pilha:

ServerLayer
UniformLayer
LoopbackLayer
VSyncLayer
LeaveLayer
StableLayer
HealLayer
InterLayer
IntraLayer
SuspectLayer
GossipOutExtLayer
GroupBottomLayer
TcpCompleteLayer

No caso da pilha que será utilizada no cliente vai ter o seguinte aspecto:

ClientLayer
ClientFilterLayer
UniformLayer
LoopbackLayer
VSyncLayer
LeaveLayer
StableLayer
HealLayer
InterLayer
IntraLayer
SuspectLayer
GossipOutExtLayer
GroupBottomLayer
TcpCompleteLayer

Temos de ter em conta que na implementação puderam vir a ser adicionadas camadas que consideremos convenientes para a resolução do problema.

7 Trabalho Futuro

A solução descrita neste artigo é bastante simplista mas garante a tolerância a faltas. As aspectos que possam ser melhorados apresentamos os seguintes:

- Alterar o acesso ao grupo de servidores por parte dos subscritores, passando a permitir que vários subscritores se liguem em simultâneo.
- Alterar de forma a quando um editor abandona um grupo, todos os subscritores do assunto que este publicava sejam notificados deste abandono.
- A actualização de servidores pode ser feita de forma mais eficiente que a que utilizamos actualmente na arquitectura descrita.
- Implementar distribuição de carga nos servidores.
- Alteração da comunicação entre o grupo servidores e subscritores, por forma a utilizar a tecnologia de RPC.

8 Conclusão

Com a evolução dos sistemas distribuídos a complexidade do desenvolvimento de aplicações tornou-se mais complexa, sendo a necessidade de tolerância a faltas um dos grandes problemas que enfrentamos. A utilização da *framework* Appia permite utilizar técnicas de tolerância a faltas já implementadas e testadas para sistemas distribuídos.

Referências

[1] Paulo Veríssimo, Luís Rodrigues, 2001. *Distributed Systems for System Architects*, Kluwer Academic Publishers.

[2] Rachid Guerraoui, Luís Rodrigues, 2005. *Introduction to Reliable Distributed Programming*, Preliminary Draft, Springer-Verlag.

[3] http://appia.di.fc.ul.pt