Jogos Multi-Utilizador em Sistemas Distribuídos e Tolerantes a Faltas

Bruno Duarte
João Gonçalves
João Silva
TFD004

DI–FCUL TFD–2005–1

Outubro 2005

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749–016 Lisboa
Portugal

Jogos Multi-Utilizador em Sistemas Distribuídos e Tolerantes a Faltas

Bruno Duarte
i30370@alunos.di.fc.ul.pt

João Gonçalves
i30337@alunos.di.fc.ul.pt

João Silva
i30327@alunos.di.fc.ul.pt

TFD004

Abstract

A distribuição de sistemas tem sido um tema bastante em foco nas últimas duas décadas, estando numa primeira fase essencialmente associada aos sistemas operativos e bases de dados. Com a democratização das redes de dados e, mais recentemente, com a expansão da Internet, este paradigma assumiu um papel cada vez mais importante na concepção das estruturas informáticas actuais.

Neste artigo abordamos o tema dos jogos multi-utilizador, em ambientes distribuídos e focando em particular a tolerância a faltas, em que utilizaremos a arquitectura de servidores replicados. O jogo utilizado para facilitar este estudo é baseado no clássico Pong, que tem a particularidade de ser jogável até um máximo de quatro jogadores.

1. Introdução

Os sistemas dos dias de hoje estão cada vez mais distribuídos, operando em ambientes dinâmicos e sujeitos a falhas, como a Internet. A implementação de sistemas distribuídos traz diversas vantagens, entre as quais a diminuição do tempo de execução, o aumento do grau de confiabilidade e disponibilidade e a inerente distribuição de uma aplicação. É com estas propriedades em mente que pretendemos realizar um jogo multi-utilizador num sistema distribuído e tolerante a faltas.

Se consideramos a complexidade e as características de ambientes de larga escala as aplicações distribuídas com requisitos de tolerância a faltas são difíceis de construir e manter.

Para aumentar a confiabilidade deste jogo iremos optar por replicar o servidor, mantendo várias cópias do mesmo num estado coerente, de maneira a que, caso um servidor falhe, uma das suas cópias consiga manter o estado do jogo, impedindo assim que o jogo sofra uma paragem abrupta. Para os jogadores esta falha passará incólume, podendo no entanto o desempenho do jogo vir a ser afectado.

2. O Jogo

O **PongFor4** é uma versão modificada do *Pong* original inventado por Ted Dabney e Larry Bryan[2] e lançado pela Atari. O PongFor4 é jogado num campo quadrado em que cada lado desse quadrado constitui uma baliza, existindo uma bola que se move pelo campo, efectuando ricochete assim que toca numa das paredes. Esta versão modificada é iniciada assim que dois jogadores iniciem uma sessão, suportando no entanto mais outros dois jogadores.

Cada jogador controla um *paddle*, que está junto à sua baliza, tendo a capacidade de movimentá-lo lateralmente ao longo da mesma. O objectivo do jogo é impedir que a bola que percorre o campo entre em contacto com a respectiva baliza. Caso a bola atinja a sua baliza é adicionada 1 golo ao seu score de golos sofridos. O jogador que sofreu um golo repõe a bola em jogo. Assim que a bola bate no *paddle* esta faz ricochete dirigindo-se para a baliza do adversário. O jogador que deixar a bola tocar mais vezes na sua baliza é o derrotado.

Sendo um jogo que suporta até quatro jogadores é conveniente que o jogo continue sempre que entre ou sai um jogador. Assim que entra um novo jogador este começa o jogo com pontuação zero sendo-lhe atribuído um *paddle*. Se já tiverem 4 jogadores em campo o novo jogador fica em espera até que um dos quatro jogadores em campo saia.

O jogo pode ter o seu fim numa de duas situações:

- Um jogador atinge a pontuação 20 sendo declarado perdedor e os restantes vencedores.
- Caso não existam jogadores suficientes em campo para jogar, ou seja, têm de existir no mínimo 2 jogadores para o jogo continuar/ser iniciado.

Como é um jogo simples e já disponibilizado pelo docente, apresenta as características necessárias para testar e optimizar uma arquitectura de tolerância a faltas num ambiente de servidores replicados.

3. Arquitectura do sistema

Sistemas centrais têm uma natural acessibilidade a informação e recursos uma vez que estes são locais. Existe uma homogeneidade de tecnologias e procedimentos o que simplifica a gestão do sistema, a consistência dos dados e a segurança do sistema é também mais acessível de manter.

Por outro lado, em sistemas distribuídos existe o potencial de expansão geográfica o que poderá facilitar a operação e acesso ao sistema e o facto deste poder ser contituido por subsistemas heterogéneos e separados geograficamente torna-o muito escalável e expansível. Outros aspectos positivos, estão relacionados com replicação de informação, execução remota, processamento paralelo distribuído e tolerância a faltas.

Para a concretização desta plataforma on-line irá ser utilizada uma arquitectura de sistemas distribuídos, mas quando se deve usar esta abordagem? Existem três aspectos muito importantes para levar em conta: quando o problema tem uma natureza descentralizada, quando as técnicas de distribuição são artefactos úteis para a solução e quando existe grande evolução da actividade e localização da organização. [3] No caso da plataforma a ser desenvolvida, as técnicas de distribuição auxiliam a eficiência e robustez do sistema, a replicação do servidor terá como objectivo principal tornar todo o sistema mais robusto e tolerante a faltas.

4. Concretização

Pretende-se criar uma plataforma de jogos em rede para multi-utilizador com servidores replicados. Os clientes (jogadores) apenas comunicam com uma das replicas do servidor, sendo que estas terão de manter um estado coerente entre elas.

4.1. Comunicação em grupo fiável

Nas aplicações distribuídas normais, interacções são varias vezes estabelecidas entre dois processos. Provavelmente, o modelo mais representativo deste tipo de internação é o clássico modelo cliente-servidor. Para cada cliente é estabelecido um canal de comunicação independente com o servidor usando protocolos de comunicação ponto-a-ponto. [4]

Com o aumento da dimensão e complexidade das aplicações distribuídas, as interacções já não estão limitadas a relações bilaterais, é importante abordarmos a questão de comunicação em grupo. Desta forma existe a possibilidade de um processo enviar uma mensagem para um grupo de processos e a certeza que todos os processos chegam a um consenso sobre a entrega da mensagem ou não.

Na concretização desta plataforma vamos usar técnicas que nos permitem uma comunicação em grupo fiável entre as várias réplicas e uma comunicação ponto-a-ponto entre o(s) cliente(s) e um dos servidores, escolhido de forma conveniente para maximizar a eficiência de todo o sistema. Para concretizar os serviços de filiação e comunicação em grupo vamos utilizar a ferramenta APPIA.

O paradigma de comunicação em grupo a implementar, com o auxílio da ferramenta em analise, é a sincronia virtual. Este paradigma tem como conceito base o facto de todos os participantes no grupo verem a mesma vista e as alterações a esta serem vistas na mesma ordem por todos os elementos do grupo.[5]

4.2. APPIA

O APPIA fornece serviços de filiação e comunicação em grupo com diferentes propriedades e tipos de ordenação irá facilitar em muito o desenvolvimento de uma aplicação tolerante a faltas.

O APPIA apresenta-se como uma ferramenta disposta em camadas que oferecem suporte à comunicação capaz de suportar a reutilização e composição de componentes.[1]

4.3. Tolerância a faltas

Uma vez que este sistema tem um grau muito elevado de complexidade, em particular devido ao facto de existirem vários canais de comunicação que poderão ter características diferentes (latência/largura de banda), irão ocorrer faltas e estas terão de ser tratadas para evitar erros e falhas no normal decorrer do jogo.

Seguidamente descreve-se as várias faltas e falhas que poderão ocorrer e como as detectar e resolver.

A falta mais comum e directamente relacionada com os servidores será o comportamento anómalo ou mesmo a falha de uma das suas réplicas. Para se poder detectar e resolver esta falta, todos os processos envolvidos no sistema, clientes e restantes réplicas, terão de ter uma vista do actual grupo de servidores. Esta vista, por forma a maximizar a eficiência, será apenas actualizada e difundida a todos os processos aquando da detecção da falta. No caso dos clientes, se a falha ocorrer na réplica à qual estão conectados, irão-se conectar a uma nova réplica. Caso uma réplica excluída ou uma nova se encontrarem em estado correcto, após a actualização do estado do jogo, estas serão adicionadas à vista e a actualização será difundida por todos os processos.

Uma vista de todos os clientes terá também de ser mantida. Aquando da detecção da falha/saída de um cliente todos os outros processos serão informados e o jogador é retirado do jogo. Neste caso, esta falha/saída apenas poderá ser detectada pela réplica à qual o cliente está conectado.

Prevê-se o caso de uma réplica com clientes falhar e neste caso, após o envio da vista das réplicas actualizada, estes terão de se conectar a uma nova réplica ou serão excluídos do jogo.

4.4. Coerência de estados

Existem alguns aspectos que, durante alguns momentos, não precisam de estar coerentes com o estado nos outros processos, um exemplo geral poderá ser a vista de réplicas de um cliente. Caso um processo cliente seja informado mais tarde do que os restantes aquando da falha de uma réplica, que não seja à qual está conectado, não existirão complicações no normal decorrer do jogo.

Existem porém alguns dados que têm obrigatoriamente de estar coerentes em todos os processos. Assim, um correcto estado do jogo será: as pontuações de todos os jogadores, o vector deslocação da bola e a posição do *paddle* de cada jogador.

5. Considerações finais

Considerando todas propriedades aqui faladas sobre sistemas distribuidos tolerantes a faltas, acreditamos que as soluções por nós previstas para a correcta manutenção deste sistema de jogo multi-utilizador irão resolver, com uma possivel alteração na jogabilidade, todos os imprevistos que possam acontecer.

O uso da plataforma de composição de protocolos APPIA irá facilitar em muito a implementação de uma comunicação em grupo fiável, através do sincronismo de vistas, entre as réplicas. A comunicação entre as réplicas e os "seus" clientes será implementada usando protocolos de comunicação ponto-a-ponto.

Devido ao facto de ainda não termos começado o desenvolvimento da aplicação, poderão surgir situações imprevistas e cuja resolução não foi aqui mencionada.

Referências

[1] *APPIA - Layered Communication Framework*. URL=http://appia.di.fc.ul.pt.

[2] *PONG Story*. URL=www.pong-story.com.

[3] Paulo Veríssimo e Luís Rodrigues. *Distributed Systems for System Architecs*. Kluwer Academic Publishers, 2001.

[4] Rachid Guerraoui e Luís Rodrigues. *Introduction to Reliable Distributed Programming*. Springer, 2005.

[5] Alexandre Pinto. *Appia Group Communication*. DI/FCUL, 2005.