Jogo distribuido multiplayer tolerante a faltas (MultiPong)

Rui Gonçalves n.º 30378 Filipe Trocato n.º 30316 Thiago Santos n.º 30404
TFD001

Abstract

Hoje em dia há uma constante preocupação com a tolerância a faltas distribuídas existentes em sistemas deste tipo. Pretende-se implementar um jogo multiplayer que se mantenha disponível mesmo quando há a ocorrência de uma falta. A construção dos novos sistemas concretiza arquitecturas que facilitam a replicação de dados obtendo assim consistência na informação descentralizada.

1 Introdução

1.1 Motivação

A ocorrência de uma falta num componente trás consequências cada vez mais graves na medida em que quando ocorre alberga custos elevados. Os sistemas tolerantes a faltas pretendem evitar/reduzir os danos causados usando diferentes técnicas tais como a replicação e a detecção de falhas.

1.2 Tema

O software a desenvolver baseia-se numa arquitectura cliente-servidor concretizada com chamadas a procedimentos remotos (RPC's) em que o cliente envia um pedido ao servidor que lhe devolve o resultado processado. O cliente usa o serviço de forma transparente não tendo conhecimento da existência de servidores replicados. O estado global do jogo deverá ser actualizado em todas as máquinas através da comunicação em grupo. A alteração do estado do jogo num cliente é enviada para o servidor, o qual replica para os outros servidores e envia para os restantes jogadores.

1.3 Estrutura

Este artigo está organizado do seguinte modo. A secção 2 revela a arquitectura do sistema. Na secção 3 explicamos alguns dos conceitos usados neste artigo; segue-se a secção 4 onde são apresentados os problemas e soluções do sistema a desenvolver. Na secção 5 apresentamos uma abordagem à resolução do problema da coordenação do jogo. A conclusão surge na secção 6 e por fim as referências consultadas na secção 7.

2 Arquitectura

Para a realização e desenvolvimento deste trabalho será utilizada a Framework do APPIA[1] como plataforma para as diversas primitivas e protocolos de suporte à comunicação. A arquitectura utilizada consiste num modelo cliente-servidor que acenta na comunicação distribuída e fiável. Os vários intervenientes do jogo podem ser divididos em dois grupos distintos, os clientes (elementos que jogam) e os servidores (elementos que fornecem o serviço). Os elementos do grupo dos servidores comunicam entre si e com cada um dos elementos do outro grupo, enquanto que os elementos do grupo dos clientes apenas comunicam com um dos servidores (primário). A comunicação entre os grupos é feita ponto-a-ponto, enquanto que o grupo dos servidores usa comunicação em grupo entre si. O servidor primário será eleito por ser o primeiro da vista. É responsável por aceitar e responder a todos os pedidos dos clientes e também por calcular o novo estado do sistema e partilhá-lo por todos os outros servidores. Esta arquitectura baseia-se no conceito de replicação passiva, explicada de seguida.

3 Conceitos

3.1 Comunicação ponto-a-ponto

Dois intervenientes comunicam entre si por mensagens UDP sem o auxílio de entidades externas.

3.2 Comunicação em grupo

Paradigma de comunicação por difusão usado na troca de mensagens entre processos cooperantes que garante certas propriedades na entrega das mensagens. Por exemplo, no protocolo de ordem total, garante-se que várias mensagens

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

recebidas por um processo correcto, serão também recebidas pela mesma ordem por todos os processos correctos do mesmo grupo (Difusão atómica).

3.3 FIFO

As mensagens enviadas por um processo devem ser recebidas por todos pela mesma ordem com que foram enviadas.

3.4 Sincronia na vista

Uma vista é uma lista que alberga um conjunto de processos correctos do mesmo grupo num dado momento. Esta vista é partilhada entre as várias réplicas e permite partilhar informação entre os processos correctos.

3.5 Replicação

Método que fornece alta disponibilidade de serviço a um sistema. Consiste na actualização constante de um estado que deverá ser global (igual em todas as máquinas). O seu uso é de grande importância nos sistemas tolerantes a faltas pois garante o serviço mesmo que uma ou mais máquinas servidor (não todas) falhem.

3.5.1 Replicação passiva

Num grupo de réplicas, uma é eleita primária e tem a responsabilidade de receber todos os pedidos e calcular/enviar os resultados. A réplica primária tem também a obrigação de comunicar o novo estado a todas as réplicas (secundárias). Este método previne a execução de código não determinista.[2]

4 Tolerância a faltas

As aplicações distribuídas estão sujeitas a falhas que poderão por em causa o normal funcionamento das mesmas. Antigamente, uma falta numa das máquinas implicaria a paragem do sistema. Os novos mecanismos protegem-no e garantem a disponibilidade do serviço nas situações mais adversas.

4.1 Tipos de faltas

As faltas relevantes para o jogo proposto são as faltas por omissão e assertivas.

[2] computações com resultados diferentes devido a ambientes de execução diferentes

4.1.1 Omissão

Este grupo abrange as faltas provocadas pela paragem de um componente (omissão) ou pela demora na entrega das mensagens (temporais).

4.1.2 Assertivas

Estas faltas podem ser sintácticas ou semânticas. No primeiro caso, surgem quando são devolvidos valores inválidos (tipos diferentes), enquanto que nas semânticas, a falha ocorre quando a informação trocada não faz sentido no domínio escolhido.

4.2 Detecção de faltas

Durante o jogo, a saída de um elemento, forçada ou não, deverá ser tratada de forma a evitar que o sistema falhe. Esta detecção usa o conceito de sincronia na vista. Na saída de um servidor, as réplicas deverão continuar a processar os pedidos, enquanto que na saída de um cliente, este deverá ser removido da vista de jogadores e a sua pontuação anulada e desactivada. Esta actualização de vistas permite que haja uma comparação entre a nova e a anterior para que se possa decidir se houve alguma saída ou entrada de máquinas para o sistema. Se o servidor primário sair todos entram em consenso de que o novo primário deverá ser o primeiro elemento da nova vista. Como esta vista é partilhada pelos clientes, estes também saberão que deverão contactar sempre o primeiro elemento, ou seja, o primário.

4.3 Entrada para o sistema

A qualquer momento pode existir uma máquina a tentar ligar-se ao sistema. Este procedimento deverá ser feito sem que se altere o desenrolar do jogo nem perturbe demasiado a jogabilidade.

4.3.1 Novo cliente

Quando o pedido de entrada de um cliente chega a um servidor, este deve inicialmente verificar quantos jogadores já estão online. Para simplificar, se já existirem 4 jogadores, o novo receberá uma mensagem a informar que a sala já está cheia e que poderá tentar ligar-se noutra ocasião. A informação deste novo cliente não será guardada. Qualquer cliente que tente ligar-se será tratado como um cliente desconhecido pois não é guardado o estado ou informação (como a pontuação) dos clientes para uso posterior.

4.3.2 Novo servidor

O servidor fica online e recebe a vista por defeito que só o contém a ele. Como o jogo pode decorrer com apenas um

servidor, este deve ter a capacidade de receber pedidos de clientes e responder aos mesmos; no entanto, se este aceitar um único cliente, e depois se for juntar a um grupo de servidores que já têm 4 clientes, surge-nos um problema de excesso de clientes. O servidor só deve então aceitar pedidos após ter tentado contactar outros servidores. Caso consiga, será incluído na vista do grupo, senão começará a tratar dos pedidos que chegarem de clientes.

4.3.3 Início de actividade

Tanto para o servidor como para o cliente, foi decidido que estes só iniciarão a troca de mensagens após a recepção de um estado. Note-se que os protocolos usados garantem a ordem total de mensagens. No instante em que uma máquina entra para a vista de um grupo, ainda não tem o mesmo estado. Como as mensagens chegam pela mesma ordem a todas as máquinas, no momento em que recebem esse estado e o guardam em memória, poderão começar a processar as novas mensagens.

5 Coordenação do jogo

Num sistema distribuído corremos o risco de várias máquinas calcularem resultados diferentes[3]. Estes resultados afectariam o desenrolar do jogo (estado) de máquina para máquina.

5.1 Estado dos elementos do jogo

Deverá existir um coordenador responsável por actualizar os jogadores de alterações ao estado do jogo (movimentos dos paddles e da bola) e também calcular o estado global do jogo. Este será o servidor primário que responderá a todos os clientes. Todos os servidores secundários funcionarão como repositórios do estado actual até que o servidor primário falhe e um deles seja o novo eleito.

5.2 Informação dos clientes

Os clientes apenas processam o estado recebido para a interface mas não o mantêm em memória. Os jogadores não devem partilhar o estado entre si mas apenas enviar as alterações do paddle para o servidor.

6 Conclusões

As soluções propostas neste artigo, têm como objectivo a introdução de mecanismos de tolerância a faltas num jogo multi-jogador permitindo que este possa ser jogado mesmo que um dos servidores falhe. A comunicação em grupo será a forma utilizada para replicar o servidor de jogo. A replicação passiva de servidores e a sincronia nas vistas são dois paradigmas que pertencem às soluções referidas anteriormente. Estas técnicas garantem que em qualquer momento, todas as réplicas têm o mesmo estado de jogo.

[3] computação não determinista

7 Referências

[1] Appia Layered Communication Framework: http://appia.di.fc.ul.pt/
[2] Paulo Veríssimo and Luís Rodrigues: Distributed System for System Architects
[3] Rachid Guerraoui and Luís Rodrigues: Introduction to Reliable Distributed Programming