Tolerância a Falhas em Jogos On-line

Fault Tolerance in On-line Games

Etelvina Pinho
Departamento de Informática
etelvina.nunes@gmail.com

Eduardo Aires
Departamento de Informática
eduardoaires@gmail.com

Paula Prata
Departamento de Informática
Instituto de Telecomunicações
Universidade da Beira Interior
6200-001 – Covilhã
pprata@di.ubi.pt

Resumo

Os jogos on-line constituem uma indústria que movimenta milhões e milhões de dólares. Apesar de um jogo não ser uma aplicação crítica no sentido tradicional do termo, a tolerância a falhas surge como um aspecto cada vez mais importante para assegurar uma alta taxa de disponibilidade. Algumas horas de inactividade de um jogo originam grandes prejuízos, quer por perda directa de jogadores nesse intervalo de tempo quer por perda da credibilidade do jogo. Neste artigo estudamos o custo da aplicação de técnicas de tolerância a falhas, como a replicação de dados e de processos a uma versão de um jogo multi-utilizador clássico, o Dungeons & Dragons. A avaliação dos

Abstract

On-line games are nowadays part of an industry that involves several million dollars in transactions. Despite games are not critical applications in a traditional way, fault tolerance as a path to achieve high availability rates starts to emerge. A few hours or even days of game downtime can represent significant losses, either by subsequent player resignation or a strong credibility decrement. This article embodies a study on the cost of fault tolerance techniques, for both data and state replication, when applied to the well-known multiplayer game Dungeons & Dragons. The evaluation of the replication cost was done using the JMeter tool to simulate several simultaneous games. We reasoned

Este trabalho foi parcialmente financiado pela FCT, Fundação para a Ciência e a Tecnologia, através do projecto MOGGY - A Browser-Based Massive Multiplayer Online Game Engine Architecture, referência PTDC/EIA/70830/2006.

resultados, feita através da simulação de testes de carga, mostra que apesar de desempenho diminuir com o número de utilizadores, essa diminuição deve-se mais ao aumento do número de utilizadores do que à replicação.

Palavras – chave: *Jogos On-line, Tolerância a Falhas, Replicação de Dados, Replicação de Processos*

out that although performance becomes smaller with the increase of connected players, that drawback is mainly related to the number of connections rather than replication itself.

Keywords: *On-line Games, Fault Tolerance, Data Replication, State Replication*

1. Introdução

Neste artigo apresentamos um estudo sobre replicação de dados e processos em jogos on-line de forma a permitir a continuidade do jogo em caso de falha. Adicionalmente é feita a análise do custo da replicação em termos do desempenho do sistema com e sem falhas.

Para este estudo foi construída uma versão tolerante a falhas do bem conhecido jogo "Dungeons & Dragons" (D&D). Na replicação da base de dados foi implementado um protocolo *lazy primary copy* (Ozsu, 2007) no qual pelo menos duas cópias da base de dados são actualizadas e no caso de uma delas falhar a outra garantirá a continuidade do jogo. Para a replicação de processos foi usado um *cluster* de servidores Web segundo um modelo de comunicação *all-to-all* (Brittain, 2007). Em caso de falha os utilizadores são redireccionados para um servidor de *backup* através de técnicas de *load balancing.* Com os dois mecanismos de replicação implementados, de forma transparente para o utilizador, foi possível preservar a qualidade do jogo em termos de tempos de resposta assim como manter a sua continuidade mesmo em caso de ocorrência de falha. Em todas as situações de falha testadas, através do crash do servidor de base de dados e/ou do crash do servidor Web, foi possível manter a continuidade do jogo, o que parece indicar que a probabilidade de uma falha levar à interrupção ou à inconsistência do jogo, na versão tolerante a falhas, é reduzida. No entanto para avaliar tal probabilidade será necessário proceder à injecção de falhas de uma forma sistemática o que esteve fora do âmbito deste trabalho.

O impacto no desempenho do jogo, dos mecanismos implementados, foi avaliado através da ferramenta JMeter (JMeter, 2009), com a simulação de múltiplos jogos em simultâneo e medindo os valores do *throughput* para situações com e sem falhas. Simulações até 300 jogadores em simultâneo mostram um custo de replicação de processos de cerca de 5% e um custo de replicação da base de dados de cerca de 30%. Os resultados mostram ainda que a

diferença entre o *throughput* com replicação e o *throughput* sem replicação se vai atenuando à medida que o número de jogadores aumenta o que significa que o custo da replicação se vai repartindo pelos vários jogadores.

O artigo tem a seguinte estrutura: nesta secção introduzimos algumas características dos jogos on-line, nomeadamente os trabalhos existentes na área da tolerância a falhas em jogos. Apresentamos ainda o jogo Dungeons & Dragons (D&D) que usamos como ponto de partida deste trabalho. Na secção 2 descrevemos a versão tolerante a falhas do D&D (FT-D&D – Fault Tolerant Dungeons & Dragons) e como foi implementada a replicação de dados e processos. Na secção 3, descrevemos o contexto experimental utilizado para testar os mecanismos implementados e medir o *throughput* do jogo para um crescente número de utilizadores através da simulação de múltiplos jogos. Na secção 4 apresentamos os resultados obtidos e finalmente na secção 5 descrevemos as conclusões e o possível trabalho futuro.

1.1. Jogos On-line e Tolerância a Falhas

Os jogos *on-line*, com possibilidade de milhares de utilizadores em simultâneo (MMOGs - *Massively multiplayer on-line games*), são aplicações cada vez mais exigentes, quer do ponto de vista das tecnologias envolvidas, como interfaces gráficas e bases de dados, quer do ponto de vista dos recursos necessários para garantir um elevado desempenho, escalabilidade, segurança e tolerância a falhas (White, 2007). Estes jogos permitem milhares de ligações simultâneas dentro do mesmo mundo virtual sendo jogados por milhões de utilizadores que exigem uma cada vez maior disponibilidade.

Os MMOGs representam um mercado muito lucrativo no sector dos jogos (Gallagher, 2009) o que leva ao constante surgimento de novas empresas. Por exemplo, o jogo World of Warcraft foi um dos impulsionadores deste crescimento ao garantir, sozinho no ano de 2006, receitas de 471 milhões de dólares (White, 2007). Em 2008, as subscrições deste jogo atingiram os 16 milhões (MMOG, 2009), sendo ainda um dos jogos que possuiu mais subscrições activas em 2009.

A maioria dos jogos *on-line* segue uma arquitectura cliente-servidor, onde os servidores são responsáveis por guardar e gerir todos os dados relacionados com o estado do jogo. Os clientes, por sua vez, são responsáveis por mostrar graficamente esses dados. Os jogos *on-line* podem ser baseados no cliente ou no *browser*. Nos baseados no cliente, como é o caso do World of Warcraft é necessário instalar o *software* do jogo no computador do utilizador. Os

jogos baseados no *browser* têm a vantagem de apenas ser necessário um computador com acesso à internet e um *browser* para aceder ao jogo. Têm ainda a vantagem de a versão mais recente do jogo estar sempre disponível. A qualidade gráfica é uma desvantagem do modelo baseado no *browser* quando comparado com o modelo baseado em cliente.

Apesar de os jogos não serem uma aplicação crítica no sentido clássico do termo, a tolerância a falhas é uma questão crucial pois uma falha, nos servidores dos jogos *on-line*, implica um efeito negativo na "experiência do jogo". Quem joga estes jogos espera que os servidores estejam permanentemente disponíveis, e que os períodos em que isso não acontece sejam reduzidos ou inexistentes. Estima-se que a falha que ocorreu no World of Warcraft em 2006 tenha custado à empresa Blizzard Entertainment cerca de $26198 por hora (Assiotis, 2006).

Hoje em dia, a escalabilidade nos MMOGs é conseguida através de um conjunto de servidores dedicados, ligados a uma rede de alta velocidade formando um *cluster* de servidores (Greene, 2008). Esta arquitectura permite aumentar a escalabilidade e a disponibilidade de um jogo através de técnicas de balanceamento de carga (*load balancing*). O nosso trabalho incide em explorar a contribuição da replicação de dados e de processos em jogos on-line para obtenção de uma maior disponibilidade em caso de falha. Apesar de usarmos mecanismos de *load balancing*, com estes apenas se pretende obter o redireccionamento dos pedidos do jogo em caso de falha. O estudo de técnicas de *load balancing* para optimizar o desempenho e a escalabilidade do jogo está fora do âmbito deste trabalho.

A replicação de dados aplicada a jogos é estudada em (Lin, 2006) usando um sistema de middleware para suporte à replicação proposto em (Lin, 2005), o MiddleSir. Naquele trabalho é analisado o impacto de diferentes protocolos de replicação no desempenho de um pequeno jogo multi-utilizador. Em (Griwodz, 2002) é proposta a replicação de estados em MOG's através da separação do tráfego baseada no nível de urgência e relevância dos elementos do jogo. Em nenhum destes casos é estudado o impacto de uma falha no desempenho do sistema. Em (Hu, 2008) é proposto um sistema de replicação de estados para MMOG's com arquitectura peer-to-peer definido através de diagramas de Voronoi. Apesar de aplicável a jogos cliente-servidor não é apresentada qualquer avaliação do impacto da replicação. Outros estudos em replicação de dados com o objectivo de obter maior escalabilidade são apresentados em (Muller, 2006) e (Ploss, 2008).

1.2. Dungeons & Dragons

O jogo *on-line* Dungeons & Dragons (D&D) originalmente conhecido como The Fantasy Game foi o primeiro *role-playing game* (RPG). Criado por Gary Gygax e Dave Arneson em 1972 foi lançado em 1974 pela Tactical Studies Rules (D&D, 2009) tendo o seu sucesso comercial levado à criação de jogos como Neverwinter Nights, Baldur Gate I & II. A última edição do jogo D&D foi lançada em 2008 sendo um jogo baseado no cliente.

Para este estudo, implementamos uma versão simplificada do D&D, a que adicionamos replicação de dados e processos e a que chamaremos Dungeons & Dragons Tolerante a Falhas (FT-D&D – Fault Tolerant Dungeons & Dragons). O FT-D&D foi baseado no jogo *on-line* Dungeons & Dragons Basic Game 2006 Edition. É um jogo muito simples em termos de interface gráfica que, não tendo a atractividade dos jogos comerciais, possui as características base de um jogo *on-line* multi-utilizador. É um jogo baseado no *browser*, com arquitectura cliente-servidor e persistência de dados. Estudamos o impacto da replicação de dados e processos em termos de *throughput* do jogo e da sua variação com o crescimento do número de utilizadores, simulando até 300 jogadores em simultâneo. Observamos que a replicação da base de dados tem um impacto muito mais significativo do que a replicação de processos no desempenho do jogo. Com o aumento do número de jogadores concluímos que o impacto da replicação (dados e processos) é cada vez menor, o que significa que o principal factor para a diminuição do desempenho é o aumento do número de jogadores. Finalmente simulamos a ocorrência de falhas quer no servidor web, quer no servidor de bases de dados estudando o custo da recuperação do jogo, isto é, a diminuição do *throughtput* quando ocorre uma falha. Observamos que, em todas as simulações realizadas, uma falha do servidor web, é transparente para o utilizador. Uma falha no servidor de base de dados implica uma pausa perceptível ao utilizador sendo necessário procurar soluções mais eficientes.

2. Dungeons & Dragons Tolerante a Falhas (FT-D&D)

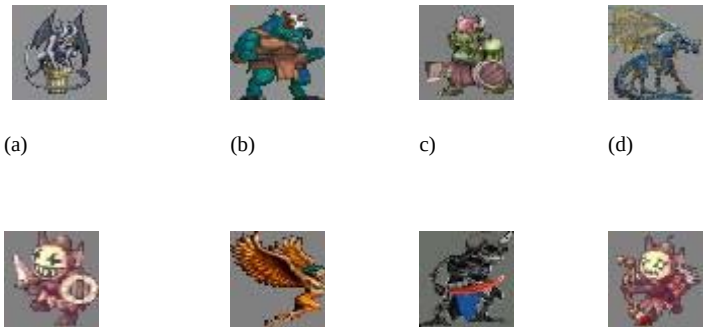
FT-D&D é um jogo de fantasia medieval, que pode ser encarado como uma forma de entretenimento em grupo, uma vez que para um jogo acontecer são necessários cinco jogadores. Cada jogador escolhe uma personagem para o representar em cada jogo. Mestre, Clérigo, Elfo, Humano e Feiticeiro são as personagens do jogo, cujas imagens com excepção do mestre são apresentadas na figura 1. As personagens possuem características distintas ao

nível da defesa, ataque ou vida, o que lhes confere uma diferente evolução. O jogador que representa a personagem Mestre é quem define a história do jogo, controlando os monstros e escolhendo os cenários. Na figura 2 podemos ver os monstros que o Mestre controla e adiciona ao jogo.

(a) Clérigo (b) Feiticeiro c) Humano (d) Elfo

Figura 1: Imagens das personagens do jogo.

O FT-D&D é um jogo com vários níveis no qual os jogos inacabados podem ser retomados, em qualquer altura, do ponto em que os deixaram, uma vez que possui persistência de dados. Para o jogo não se tornar monótono e para promover uma maior interacção entre jogadores, foi criado um *chat* que permite a comunicação de forma rápida e intuitiva. A figura 3 mostra um dos cenários do jogo, onde podemos ver um tabuleiro (ao centro) com as personagens e alguns monstros, as características do jogador (à esquerda) e a informação de quais os jogadores e respectiva personagem (à direita). Em baixo vemos a área de texto do *chat* que permite a comunicação entre os jogadores.



(a) (b) c) (d)

(e) (f) g) (h)

Figura 2: Imagens dos monstros do jogo.

Figura 3 – Cenário após o mestre configurar o início do jogo.

Este jogo foi desenvolvido no sistema operativo Windows e implementado em Java, JSP, XHTML, JavaScript, JSP Scriptlet. Os dados do jogo e de login são armazenados num servidor de base de dados MySQL. O servidor web é o servidor "Web Apache Tomcat".

2.1. Replicação de Dados

A replicação de dados no jogo FT-D&D foi realizada entre dois servidores de base de dados seguindo o protocolo de replicação *Lazy primary copy*, (Gray, 1996) e (Lin, 2006). Neste protocolo são usados dois ou mais servidores de bases de dados, um como servidor primário, ou *master*, e os outros como servidores secundários ou *slaves*. Todos os pedidos efectuados pelos clientes são direccionados para a réplica primária (*master*). Após a actualização da base de dados ser efectuada com sucesso, é enviada aos clientes a notificação de *commit* e de seguida os dados actualizados são transmitidos para o *slave* (ver figura 4).

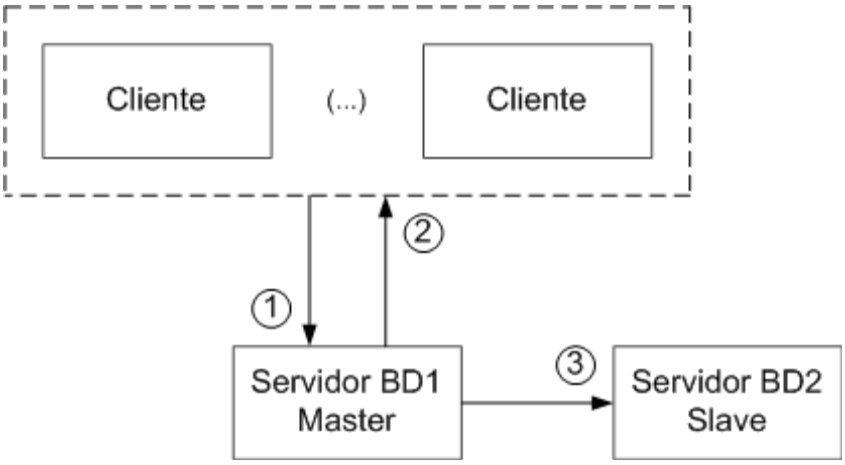


Figura 4: Esquema da replicação de dados no jogo FT-D&D.

Com o servidor de bases de dados MySQL a replicação pode ser configurada segundo três modelos (Schwartz, 2008), *master-master* modo activo-activo, *master-master* modo activo-passivo e *master-slave*. No modelo *master-slave*, as actualizações efectuadas no *master* são posteriormente transferidas para os *slaves*. Neste modelo não é possível a um *slave* assumir o papel de *master*, caso este último falhe. É um modelo simples, apropriado por exemplo para possuir uma base de dados de *backup* ou para distribuir pedidos de leitura. O modelo *master-master* modo activo-activo permite que dois servidores sejam configurados simultaneamente como *masters* e como *slaves*. O modelo permite escrever directamente em ambos os servidores, sendo necessário lidar com possíveis conflitos. Finalmente, o modelo *master-master* modo activo-passivo permite a troca de papéis entre o *master* e o *slave*. Este foi o modelo implementado uma vez que permite que os pedidos dos clientes sejam redireccionados para a réplica secundária (*slave*) no caso de falha do *master*. O *slave* terá sido entretanto reconfigurado para se comportar como *master* (ver figura 5). Quando o antigo *master* recupera da falha é transformado em *slave*, passando a receber todas as actualizações efectuadas no novo *master*. A alteração de *master* para *slave* é realizada através do jogo on-line no momento em que é detectada uma falha no servidor de base de dados. Este protocolo comporta-se de forma assíncrona, sendo os dados actualizados no servidor *master* e replicados para o *slave*. Este tipo de actualização permite que se uma falha ocorrer num dos servidores os outros não serão afectados.

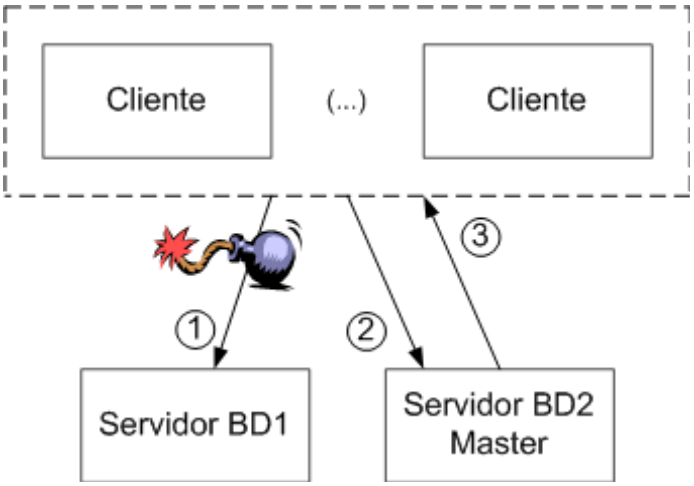


Figura 5: Representação da mudança de servidor de base de dados no jogo FT-D&D.

O ponto crítico dos protocolos assíncronos consiste em ocorrer uma falha no master imediatamente após a conclusão de uma jogada mas antes de ser concluída a replicação. Tal situação nunca ocorreu nos testes efectuados, o que nos permite supor que a probabilidade de ocorrência será baixa.

Outro protocolo possível seria a replicação *eager* (Wiesmann, 2005) em que a actualização dos vários servidores é englobada numa única transacção. Neste caso o problema seria o facto de os dados só se tornarem definitivos após todas as bases de dados terem sido alteradas. No caso da falha de uma base de dados, durante uma transacção de actualização, seria feito um *rollback* em todas as réplicas o que comprometeria a continuidade do jogo.

2.2. Replicação de processos

A replicação de processos no jogo FT-D&D foi implementada num *cluster* de servidores "Web Apache Tomcat", composto por duas instâncias. A definição dos membros do *cluster* pode ser realizada de forma estática ou dinâmica sendo esta última a utilizada. Na forma estática, é necessário definir em todas as instâncias o IP dos membros que pertencem ao *cluster*. É uma solução para redes em que a comunicação *multicast* não é possível e para *clusters* de dimensão reduzida e/ou IP fixo. Mostra-se inviável em *clusters* em que é requerida uma elevada disponibilidade, pois para a remoção ou inserção de um membro no cluster é necessário alterar os ficheiros de configuração de todos os membros. A forma dinâmica implementada permite a adição/remoção de instâncias do *cluster* sem que para isso seja necessário alterar a configuração dos membros activos. Tal é possível devido à auto-

descoberta dos membros do *cluster* que é realizada através da comunicação *multicast*. Esta é assim a solução ideal para *clusters* de grande dimensão e/ou com IP dinâmico. A replicação de processos escolhida, segue o modelo *all-to-all* ou seja, cada membro do cluster envia mensagens com os dados de sessão para os membros activos (Brittain, 2007).

A troca de mensagens entre os membros do *cluster* pode ser realizada de forma síncrona ou assíncrona podendo ainda ter ou não um *acknowledged*. O modo assíncrono sem *acknowledged* foi o implementado uma vez que há uma redução do tráfego na rede e portanto um melhor desempenho na replicação dos dados de sessão. Neste modo os membros secundários do *cluster* apenas recebem as actualizações após o novo estado ter sido enviado ao cliente (ver figura 6). O cliente tem assim um acesso mais rápido ao novo estado do jogo do que com comunicação síncrona. No modo síncrono o estado do jogo é replicado para os membros do *cluster* antes do novo estado ser enviado ao cliente.

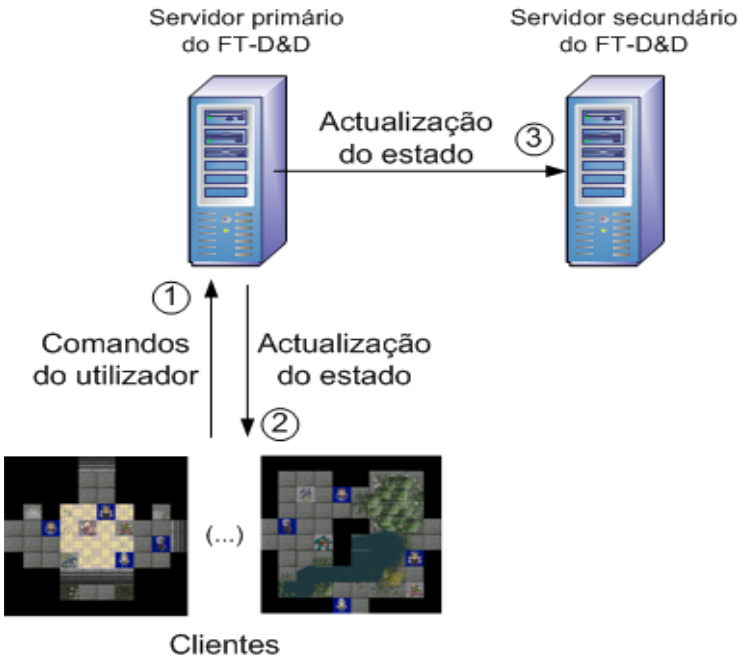


Figura 6: Esquema da replicação assíncrona no jogo FT-D&D

O envio das mensagens de sessão, entre os membros do *cluster*, tem necessariamente de seguir a ordem pela qual as acções ocorreram no jogo. Para que a ordem seja garantida foram sincronizados os relógios dos membros do *cluster* através do *Network Time Protocol* (NTP) (Mills, 1989) e (Brittain, 2007).

Finalmente para a replicação de processos ser transparente para os clientes, foi criado um *Domain Name System* (DNS) e implementado um sistema de *load balancing*. Para implementar o *load balancing* foi utilizado o "apache server httpd" com o modulo jk. Este módulo utiliza o conceito de *worker*. Cada *worker* corresponde a um membro do *cluster*. Assim sendo, é possível definir uma instância principal e várias instâncias secundárias. Neste caso em particular, a instância secundária foi configurada para receber pedidos apenas no caso da instância primária falhar. Quando a instância primária recupera da falha todos os novos pedidos e os em execução na instância secundária, são redireccionados para a instância primária. Assim, ao ter sessões permanentemente replicadas a probabilidade de ocorrer perda de dados será baixa. Podemos considerar como ponto crítico a situação em que o servidor web principal falha após a actualização do estado do jogo mas antes de a replicação ser efectuada. Também neste caso essa situação nunca ocorreu.

3. Ambiente experimental

Para a medição dos tempos de execução de vários jogos em simultâneo, foi utilizada a ferramenta JMeter. O JMeter foi desenhado para realizar testes de carga, e medir o desempenho de aplicações *web*. É uma ferramenta *multithreading* que permite uma amostragem simultânea de diferentes funções, separadas em *thread groups* (JMeter, 2009). Para a simulação dos jogos no JMeter, é necessário guardar a sequência de todas as jogadas de um jogo. Para isso, foi utilizado um *proxy* disponibilizado pelo próprio JMeter, que capturou todos os pedidos realizados ao servidor do jogo *on-line* FT-D&D. Para a captura de pedidos foram utilizados cinco *browser´s*, devidamente configurados para a utilização de um *proxy*, em que cada um representa um jogador. Cada jogo foi guardado no JMeter, em grupos distintos, o que tornou possível a simulação de vários jogos em simultâneo. Foram medidos os tempos de execução para 1, 15, 30 e 60 jogos em simultâneo, isto é, 5, 75, 150 e 300 utilizadores.

A arquitectura física utilizada para a simulação dos vários jogos, quando são implementados todos os mecanismos de replicação, foi a seguinte: 2 servidores web; 2 servidores de bases de dados; 1 servidor com o *load balancer* e DNS (ver figura 7); 4 servidores com o JMeter. Todos os servidores possuem características iguais, ou seja, processador Intel Pentium D a 3GHZ e 1Gb de RAM, sistema operativo Windows Server 2003, com uma rede *directed broadcast* a 100 Mbs.

Todas as simulações passam obrigatoriamente pelo servidor de DNS e de *load balancer* que reencaminha o pedido para o servidor web. Foram necessários 4 servidores com a ferramenta JMeter para simular 60 jogos em simultâneo, devido a limitações de memória na máquina virtual do Java. Foi necessário distribuir por cada JMeter, 15 jogos iguais na sequência de jogadas mas com diferente identificador. Esse conjunto de 15 jogos tentou ser uma amostra representativa de uma situação real. Um terço corresponde a jogos de baixo nível de complexidade, um terço de nível médio e os restantes de nível elevado.

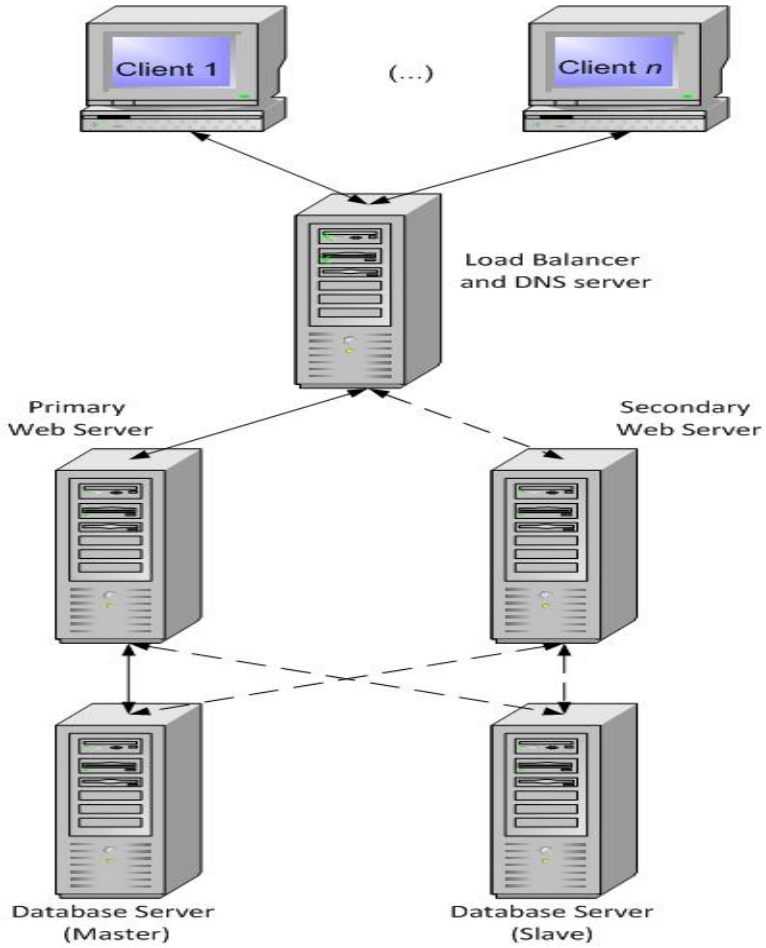


Figura 7: Arquitectura implementada para o jogo FT-D&D.

É importante notar que a arquitectura implementada tem dois pontos críticos, o servidor de DNS e o servidor de *load balancing*. Uma falha num destes servidores comprometeria todo o jogo. Estes pontos críticos poderão ser ultrapassados através da replicação dos servidores.

4. Resultados

Para a simulação dos jogos foram consideradas sete situações para o servidor do jogo, correspondendo cada uma delas a uma linha de resultados na tabela I:

1) Sem replicação – não existe replicação de dados nem de processos;

2) Replicação de processos – existe apenas replicação de processos com duas instâncias do Tomcat em máquinas distintas e uma máquina com o DNS e o "load balancing";

3) Replicação de dados – o servidor de base de dados principal é configurado para efectuar a replicação, um segundo servidor desempenha o papel de secundário;

4) Replicação de dados e de processos – situação 2 mais 3;

5) Falha do servidor web – na situação 2 (apenas com replicação de processos) faz-se falhar o servidor web principal (matando o processo). Após a falha, os pedidos são redireccionados para o servidor secundário que possui informação sobre as sessões activas.

6) Falha do servidor de base de dados – na situação 3 (apenas com replicação de dados) é provocada uma falha no servidor de base de dados principal.

7) Falha do servidor de base de dados e do servidor web – na situação 4 (com replicação de dados e de processos) é provocada uma falha no servidor web principal e posteriormente no servidor de base de dados principal (master).

Para as situações de falhas (5, 6 e 7), os tempos apurados incluem o tempo do redireccionamento para os servidores secundários o que assegura a continuidade do jogo. Não se incluem os tempos para recuperar os processos que falharam e a sua reintegração na arquitectura do jogo por estes não terem impacto significativo no desempenho do jogo.

Na figura 8 é apresentado um gráfico com o *throughput* (valor médio do número de pedidos processados por segundo) obtido para 1, 15, 30 e 60 jogos, construído a partir dos dados apresentados na tabela I. Nesta tabela são apresentados os valores do *throughput* em valor absoluto e em percentagem (colunas à direita) a qual foi calculada tomando como base o valor obtido sem qualquer replicação.

Observamos que a capacidade de resposta para as situações sem replicação e com replicação de processos não é muito diferente, isto é, o custo da replicação do processo é pouco significativo. A capacidade de resposta do servidor, *throughput*, com replicação de processos passa para 95% quando consideramos 60 jogos (sendo de 97% para 15 jogos e 94% para 30

jogos). O custo da replicação de dados é mais significativo. Com 15 jogos obtém-se 74% de desempenho, para 30 jogos 70%, e finalmente para os 60 jogos o desempenho é 67% do obtido sem replicação. Finalmente na replicação de dados e processos, como seria de esperar, há um acumular da perda de desempenho. Para 60 jogos, isto é 300 jogadores, a replicação de dados e processos tem um desempenho de 62% (isto é, um custo de 38%).

Verifica-se que com o aumento do número de jogos há uma perda de desempenho mas essa perda vai sendo cada vez mais atenuada, isto é, a distância entre o *throughput* para o jogo sem replicação e o *throughput* obtido com replicação vai sendo cada vez menor. O custo da replicação vai sendo repartido pelos vários jogadores como pode observar-se pela aproximação das linhas do gráfico. Podemos concluir que o aumento do custo se deve essencialmente ao aumento do número de pedidos.

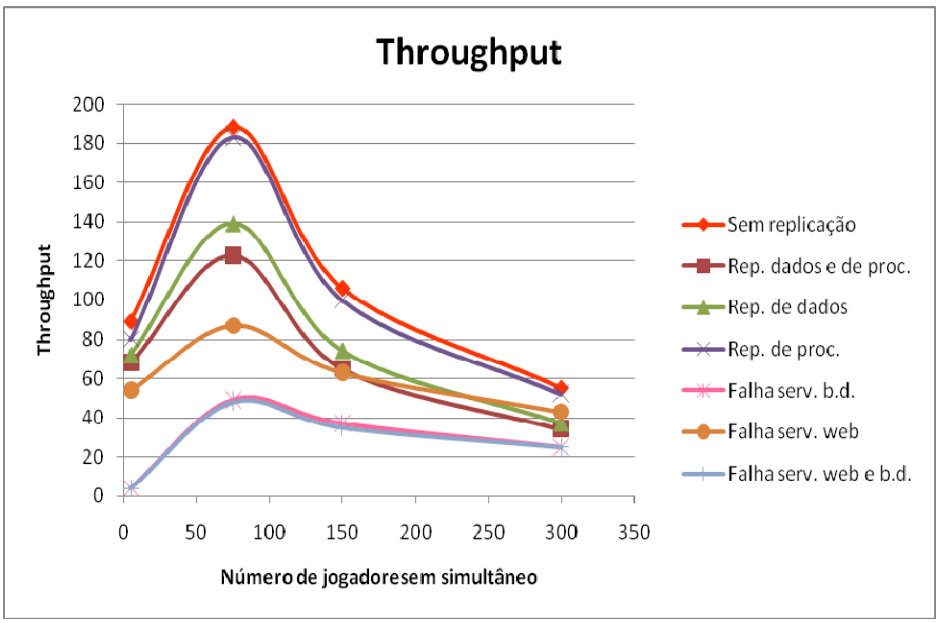


Figura 8 – *Throughput* para 1, 15, 30, e 60 jogos em simultâneo

Considerando os casos com simulação de falhas, podemos observar que o custo da ocorrência de falhas é bastante mais alto do que o custo de apenas a replicação. Iremos somente analisar os valores para 60 jogos uma vez que são mais significativos. Se, no período de tempo considerado para a simulação, ocorrer uma falha no servidor Web (caso 5) o *throughput* desce para 78%, e uma falha no servidor de base de dados (caso 6) provoca um decréscimo para 45%. Quando ambas as falhas são simuladas (caso 7) o *throughput* permanece nos 45%. O impacto da falha no servidor Web, que ocorre primeiro, é praticamente nulo. O tempo gasto

na troca de servidor Web parece ser compensado pelo tempo poupado ao não haver comunicação dos dados para a réplica. Devemos notar que o servidor Web que falhou não foi recuperado até ao final da simulação considerada.

Tabela I - *Throughput* em valor absoluto e em percentagem para 1, 15, 30, e 60 jogos em simultâneo.

	Throughput (número de pedidos por segundo)				**Throughput (%)**			
	1 jogo	**15 jogos**	**30 jogos**	**60 jogos**	**1 jogo**	**15 jogos**	**30 jogos**	**60 jogos**
1. Sem replicação	89	188	106	55	*100*	*100*	*100*	*100*
2. Replicação de processos	80	183	100	52	*90*	*97*	*94*	*95*
3. Replicação de base de dados	72	139	74	37	*81*	*74*	*70*	*67*
4. Rep. de bd e processos	68	123	65	34	*76*	*65*	*61*	*62*
Simulação de falhas								
5. Falha servidor Web	54	87	63	43	*61*	*46*	*59*	*78*
6. Falha servidor base de dados	4	49	37	25	*4*	*26*	*35*	*45*
7. Falha servidor bd/Web	4	48	35	25	*4*	*26*	*33*	*45*

A falha da base de dados tem um impacto significativo, o que se deve em grande parte ao facto da detecção da falha e mudança de servidor de base de dados estar a ser feita ao nível do jogo. Estes tempos poderão ser melhorados com os mecanismos de replicação e *clustering* disponíveis nas versões mais recentes dos servidores de bases de dados. As simulações efectuadas tiveram uma duração de alguns minutos, correspondendo ao completar dos jogos considerados. A probabilidade de ocorrer uma falha num intervalo de tempo tão pequeno é praticamente nula. Assim, podemos considerar que o custo da tolerância a falhas implementada corresponde apenas ao custo da replicação.

5. Conclusão

Neste artigo apresentamos um estudo do impacto da introdução de mecanismos de tolerância a falhas, como a replicação de dados e processos em jogos on-line. Foi desenvolvida e estudada uma versão tolerante a falhas do jogo Dungeons & Dragons com um servidor redundante de base de dados e um cluster com dois servidores web.

Concluímos que o impacto da replicação de processos é diminuto causando um decréscimo no desempenho de cerca de 5%. A replicação da base de dados tem um impacto mais significativo passando a capacidade de resposta do servidor para 67%, isto é um decréscimo 33%. Com o aumento do número de jogadores o impacto da replicação é cada vez menor, o que significa que o principal factor para a diminuição do desempenho é o aumento do número de jogadores.

Neste estudo não foi avaliada a taxa de cobertura dos mecanismos de tolerância a falhas implementados, isto é, não estimámos a probabilidade de uma falha levar à inconsistência do jogo. Tal estudo poderá ser feito no futuro através da injecção de falhas. Adicionalmente pretendemos explorar soluções para remover os pontos críticos relacionados com o facto de uma falha ocorrer após uma operação de actualização, mas antes da replicação. A solução óbvia que seria o uso de protocolos síncronos não é viável, mas deverão ser estudados protocolos semi-síncronos. Finalmente, tendo em conta que neste trabalho a detecção de falha e a mudança de servidor foram feitas no código do jogo, pretendemos estudar, como trabalho futuro, a utilização de um *cluster* de base de dados com detecção automática de falha e respectiva mudança de servidor.

Referencias bibliográficas

Assiotis, Marios and Tzanov, Velin (2006) A distributed architecture for MMORPG, NetGames '06: Proc. of 5th ACM SIGCOMM workshop on Network and system support for games, (4).

Brittain, Jason, Darwin, Ian (2007), Tomcat: The Definitive Guide, 2nd Ed., O'Reilly Media Inc.

D&D (2009), História do jogo *on-line* Dungeons & Dragons, http://www.wizards.com/dnd/DnDArchives_History.asp, data de acesso: Julho de 2009.

Gallagher, Michael (2009), 2009 state of the industry address – Delivered by Michael Gallagher president and CEO, ESA http://www.theesa.com/, data de acesso: Julho de 2009.

Gray, J. and Helland, P. and O'Neil, P. and Shasha, D. (1996) The Dangers of Replication and a Solution, SIGMOD'96.

Greene, Robert (2008), Advanced Data Management for MMOG, Versant Corp.

Griwodz, Carsten (2002), State replication for multiplayer games, NetGames '02: Proceedings of the 1st workshop on Network and system support for games, (29-35).

Hu, Shun-Yun, Chang, Shao-Chen, Jiang, Jehn-Ruey (2008) Voronoi State Management for Peer-to-Peer Massively Multiplayer Online Games, Consumer Communications and Networking Conference, 1134-1138.

JMetter (2009), Apache JMeter http://jakarta.apache.org/jmeter, data de acesso: Julho de 2009.

Lin, Yi and Kemme, Bettina and Patino-Martinez, Marta and Jimenez-Peris, Ricardo (2006) Applying database replication to multi-player online games, NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games.

Lin, Yi, Kemme , Bettina, Patino-Martínez , Marta, and Jiménez-Peris, Ricardo (2005). Middleware based data replication providing snapshot isolation. In SIGMOD '05: Proc. of the 2005 ACM SIGMOD int'l conf. on Management of data, pp. 419–430, New York, NY, USA, ACM.

Mills, D. L., (1989) Internet time synchronization: The network time protocol, URL: http://www.ece.udel.edu/~mills/ntp/html/index.html, data de acesso: Julho de 2009.

MMOG (2009), Total MMOG Active Subscriptions. http://www.mmogchart.com/Chart4.html, data de acesso: Julho de 2009.

Muller, Jens, Gossling, Andreas and Gorlatch, Sergei (2006), On correctness of scalable multi-server state replication in online games. In NetGames '06: Proc. of 5th ACM SIGCOMM workshop on Network and system support for games, page 21, New York, NY, USA. ACM.

Ozsu, M. Tamer (2007), Principles of Distributed Database Systems. Prentice Hall Press, Upper Saddle River, NJ, USA, (657 pages).

Ploss, Alexander, Wichmann , Stefan, Glinka , Frank, and Gorlatch, Sergei (2008). From a single to multi-server online game: a quake 3 case study using rtf. In ACE'08: Proc. of the 2008 Int'l Conf. on Advances in Computer Entertainment Technology, pages 83–90, New York, NY, USA, ACM.

Schwartz, Baron, Zaitsev , Peter, Tkachenko, Vadim, Zawodny , Jeremy, Lentz, Arjen and Balling , Derek J. (2008). High performance mysql, 2nd edition. O'Reilly, 2008.

White, Walker, Koch, Christoph, Gupta, Nitin, Gehrke, Johannes, Demers, Alan (2007) Database research opportunities in computer games, SIGMOD Rec., 36, 3, (7-13).

Wiesmann, M. (2005), Comparison of database replication technique based on total order broadcast, IEEE Trans. on Knowl. and Data Eng.