

Load Balancing on an Interactive Multiplayer Game Server

Daniel Cordeiro^{1,*}, Alfredo Goldman¹, and Dilma da Silva²

¹ Department of Computer Science, University of São Paulo
`danielc,gold@ime.usp.br`

² Advanced Operating System Group, IBM T. J. Watson Research Center
`dilma@watson.ibm.com`

Abstract. In this work, we investigate the impact of issues related to performance, parallelization, and scalability of interactive, multiplayer games. Particularly, we study and extend the game QuakeWorld, made publicly available by *id Software* under GPL license. We have created a new parallelization model for Quake's distributed simulation and implemented this model in QuakeWorld server. We implemented the model adapting the QuakeWorld server in order to allow a better management of the generated workload. We present in this paper our experimental results on SMP computers.

1 Introduction

Game developers recently started a deep discussion about the uses of multiple processors in computer games. The new generation of video game consoles provides a multicore processor environment, but the current game developing technology does not use the full potential of the new video game consoles yet.

Abdelkhalek et al. [1,2,3] started an investigation of the behavior of interactive, multiplayer games in multi-processed computers. They characterized the computing needs and proposed a parallelization model for QuakeWorld, an important computer game optimized for multiplayer games.

The results presented by Abdelkhalek showed that his multithreaded server implementation suffered from performance problems because of their misuse of game semantics in order to create a lock strategy and because of the use of a static division of work between the available processors.

We present a dynamic load balancing and scheduling mechanism that utilizes the semantics of the simulation executed by the server. The performance analysis shows that we are able to improve the parallelism rate from 40% obtained by Abdelkhalek to 55% of the total execution time.

This paper is organized as follows. Section 2 gives an overview of the QuakeWorld server. Section 3 describes previous efforts in parallelizing the Quake server. Section 4 presents the proposed parallelization model for Quake and Section 5 presents the performance analysis. Section 6 presents our conclusions.

* The author would like to thank The State of São Paulo Research Foundation (FAPESP, grant no. 03/10064-4) for the financial support.

2 Quake Server

The Quake game is an interactive, multiplayer action game developed and distributed by *id Software* [4]. Its release in 1996 was an important mark in the game industry because it was the first time that a game was developed using three-dimensional models in the simulation and graphics engines. Later that year, QuakeWorld was released with enhancements for Internet games¹.

The Quake multiplayer game follows the client-server architecture. One game session has a single, centralized server that is responsible for the execution of all physics simulations, for evolving the game model (that gives the semantics of the game to the simulation), and for the propagation of state modifications to all connected clients. Up to thirty-two clients can join an open game session. It is the client responsibility to collect and send the input events from the player to the server and to do the graphics rendering of the current state of the simulation.

2.1 Quake Server Architecture

The Quake server is a single process, event-driven program. The server-side simulation consists of execution of frames. The frame task is composed by three distinct stages: updating the world physics, receiving and processing events sent by clients and creating and sending replies to all active players.

During the simulation of the world physics, each solid game entity (players, bullets, blocks, etc.) is simulated by the engine. The engine simulates the effects of gravity, velocity, acceleration, etc. The request and response processing is the more computational intensive phase. It reads all messages from the server socket, simulates the events sent in the message (movement commands, jump command, text messages, etc.), applies the command to the game state and replies to the clients with the changes in the game state. Only active clients, i.e. clients that sent a message in this frame receives a reply.

3 Multithreaded Version

Abdelkhalek et al. started an effort to characterize the behavior and performance of the original version of Quake [1]. They increased the limit of players from the original 32 to 90 simultaneous players. Their experiments showed that the incoming bandwidth is practically constant and is low (a few KBytes/s) and the outgoing bandwidth depends on the number of simultaneous clients, but does not exceed a few KBytes/s. The actual performance limitation was the processing power bottleneck. With more than 90 simultaneous users, server performance started to degrade.

In his following works [2,3], Abdelkhalek presented a multithreaded version of the Quake server. In order to avoid semantic and correctness errors that could

¹ We will use the names “Quake” and “QuakeWorld” interchangeably in this text to refer this new improved version.