Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: ees.elsevier.com/entcom

Investigating informative performance metrics for a multicore game world server [☆]

James Munro, Kofi Appiah, Patrick Dickinson [*]

School of Computer Science, University of Lincoln, Lincoln LN6 7TS, UK

A R T I C L E   I N F O

Article history:
Received 17 December 2012
Revised 14 August 2013
Accepted 26 October 2013
Available online 11 November 2013

Keywords:
Multi-core games server
Performance evaluation
Game server metrics

A B S T R A C T

Many real-time game world servers run on stand-alone PCs, such that user performance is bound to fairly modest hardware configurations. Studies of multicore architectures to optimize such servers are sparse, and evaluations typically involve the use of one or two arbitrary performance metrics. However, the behavior of game servers is complex and the interpretation of metrics, particularly in the case of parallel implementations, is not straightforward.

Our initial interest is in efficient load-balancing of multicore game engines. However, the focus of this paper is on performance metrics: starting with proposed metrics from other works, we investigate their effectiveness and inter-relationships, propose new variants, and discuss how they can be used in combination to gain a better understanding of actual performance.

The use of metrics to inform the design and optimization of game software has gained recent interest from academics and practitioners alike: we conclude to show, by example, how server metrics can be directly connected with game semantics, and used to predict the impact of game design changes on server performance.

1. Introduction

Multiplayer games range from the technologically simple, to sophisticated endeavors such as Massively Multiplayer Online Role-playing Games (MMORPGs). The concept of a client–server architecture is ubiquitous: in the case of MMORPGs, expansive environments are hosted on bespoke server configurations which facilitate huge numbers of users. For example, by 2007 the game *EVE Online* had recorded over one million unique players since its launch in 2003 [1]. Whilst a significant amount of research has investigated the use of distributed architectures to support large-scale game servers (e.g., [2–8]), this type of server setup is exceptional.

Many games allow players to create their own stand-alone *ad-hoc* servers which service smaller game worlds with tens rather than thousands of players. These servers run on standard consumer equipment, and performance is (unsurprisingly) closely bound to processing power [9]. Single machine servers represent a major part of the currently available multiplayer online gaming service, and are common for *first person shooters* games which involve fast-paced interactive gameplay and real-time simulation. Player experience for this game type is particularly sensitive to degradation in performance, in the order of milliseconds [10,11], and so server optimization represents an ongoing challenge for developers. It is therefore surprising that relatively little work has been directed at optimizing stand-alone servers to utilize the parallel processing architecture of multicore CPUs. Game metrics have attracted recent academic interest (e.g., [12,13]), and also interest from industry where they are perceived as a valuable tool for design, balancing, and optimization. As Abdelkhalek et al. note [9], benchmarking methods for interactive game servers are driven by somewhat different considerations from scientific processing: useful performance evaluation should reflect user experience in some way. Again, little work has yet considered suitable server-side metrics for the analysis of real-time multicore game engines.

1.1. Motivation

The starting point for our work is an existing server design proposed by Cordeiro et al. [14], implemented using *id software's QuakeWorld* game server. Cordeiro's work uses spatial partitioning to divide entity processing into discrete non-intersecting work packages which can executed in parallel (details of the architecture are given in Section 2.3). Our initial interest is in load balancing, and optimizing the distribution of work packages across hardware threads; however, a survey of current work in this area reveals that the use of performance metrics is not standardized, making it difficult to compare algorithms. Moreover, a single metric is not in itself entirely informative, and often leaves questions remaining

[☆] This paper has been recommended for acceptance by Minhua Eunice Ma.
[*] Corresponding author. Tel.: +44 1522886946.
E-mail address: pdickinson@lincoln.ac.uk (P. Dickinson).

1875-9521/$ - see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.entcom.2013.10.001

about the underlying processes. The measurement of performance of a multicore server thus becomes our primary interest, such that the motivations for our study are:

1. To investigate the relationship (if any) between currently used server-side performance metrics.
2. To determine which metric, or set of metrics, provide the most informative analysis of performance.
3. As a secondary motivation, we are interested in the impact that the choice of load balancing algorithm has on performance in Cordeiro et al.'s architecture: this provides a context for points 1 and 2.

As mentioned, useful performance evaluation should reflect player experience in some way. In terms of perceived responsiveness, experience is a function of several factors of which server performance is just one. Others include data transmission latency, client-side performance, and also game play context: for example, the affects of latency on player experience have been well-studied (e.g., [11,15,16]). A proper analysis of perceived responsiveness encompasses all these factors, is context dependent, and lies outside the scope of the work presented here. Our focus is specifically on identifying meaningful comparators for multicore server architectures, which may be used to quantify performance and independently optimize design. Nevertheless, our metrics do relate directly to player experience. For example, we will use server throughput, which is a direct measure of the number of connected clients that can be processed concurrently, and so has a direct effect on experience.

1.2. Contributions

Our study takes the form of a set of empirical investigations into the performance of different simple load balancing strategies used in conjunction with Cordeiro et al.'s *QuakeWorld* server [14]. These experiments are primarily constructed to investigate the response of different metrics. Building on our preliminary results, presented in [17], the contributions of this paper are:

1. We evaluate the effectiveness of a range of server-side metrics including frames per second, server throughput, thread wait time, and accumulated thread work load. We present conclusions concerning their inter-relationships and effectiveness, and which are most useful in analyzing performance. A study of performance metrics in the context of multicore game servers has not previously been conducted, and is of immediate use to developers working on stand-alone game server applications.
2. In relationship to Cordeiro et al.'s architecture [14], we show by example how metrics can be used to estimate the effect of game design changes on server performance.
3. We investigate the effects of different load balancing algorithms on server performance. We use only simple balancing techniques, but these are still able to characterize the importance of effective thread balancing in Cordeiro et al.'s system. We further investigate how these results scale across varying numbers of CPU cores, ranging from one (serial) to six concurrent hardware threads, using our metrics.

Whilst we use a specific architecture and game engine to conduct our experiments, our results are easily generalized. The proposed metrics are low-level statistics which describe the performance of workgroups processed on hardware threads: these are thus independent of the workgroup allocation strategy, and equally applicable to any multicore game server design. Furthermore, the lockless server design which we employ [14] is based on the semantic constraints of objects moving in a physical simulation. This design may therefore be transposed to any functionally comparable game engine (e.g., *first person shooter*, or game which simulates a physical world).

1.3. The structure of this paper

The rest of this paper is presented as follows. Section 2 reviews the current literature regarding parallel and concurrent processing architectures and metrics in game engines, specifically server-side, and concludes with a description of the *QuakeWorld* server, and a detailed description of the parallel implementation presented by Cordeiro et al. Section 3 proceeds to describe our experimental setup, and is followed by Sections 4–7 which present our experimental work and discussions of performance metrics. We conclude with a discussion of our results, and motivate some conclusions regarding the use of server-side metrics, and load-balancing strategies for stand-alone multicore game servers.

2. Background and related work

Whilst relatively little work has addressed the evaluation of multicore game servers, there has been considerable wider interest in the use of concurrent architectures to optimize game software. Aspects of client-side processing have been addressed by Gildea [18], who attempted to adapt the *Quake III* client to support parallel execution (with limited success). He identified the difficulty in reconstructing concurrent processing threads which access shared memory. The use of GPUs to implement concurrent graphics processing is well established. Their potential for use in non-graphical processing in game engines has also been investigated [19–23].

Our interest lies specifically in the optimization of game servers. A number of studies have considered distributed architectures: Bharambe et al. [24] succeeded in scaling the *Quake II* engine over many server nodes, supporting hundreds of players. A study by Ploss et al. [25] parallelized the *Quake III* server using a purpose-built scalable grid framework. A number of other studies ([3,4,6–8]) have dealt with distributing game state across multiple nodes.

2.1. Optimizing a stand alone server

Practical considerations dictate that *ad hoc* servers are implemented on stand-alone machines; however, relatively little work has investigated the implementation, optimization, and benchmarking of appropriate parallel architectures. As mentioned, Abdelkhalek et al. [9] analyzed the performance of the standard sequential *QuakeWorld* server, empirically determining an approximately linear relationship between processing overhead and the number of players. They discussed the difficulty of meaningful benchmarking: noting the functional similarity with online transaction processing, they propose the use of server throughput and CPU idle time, as performance metrics.

In further work, Abdelkhalek and Bilas [26] implemented a parallel version of the *QuakeWorld* server. The response processing and reply phases were processed by concurrent threads running on separate cores of a quadcore CPU. Parallel execution was achieved by assigning each player permanently to a specific thread; however, memory synchronization was a limiting factor, and the resolution of lock contentions represents up to 35% of total execution time. An analysis showed that peak response occurs with around 25% more players attached than the serial version, which is a significant improvement. In this work, Abdelkhalek and Bilas use only response rate and aggregated thread workload to analyze performance: we will show in our experiments that these alone are not sufficient to fully understand the behavior of