Load Balancing Techniques for Massively Multiplayer Online Games: A Survey
Sarmad A. Abdulazeez, Abdennour El Rhalibi, Madjid Merabti and Dhiya Al-Jumeily
School of Computing and Mathematical Sciences,
Liverpool John Moores University,
Liverpool, L3 3AF, UK
S.A.Abdulazeez@2013.ljm.ac.uk, {A.Elhalibi, M.Merabti, D.Aljumeily}@ljmu.ac.uk

Abstract— In recent years, there has been an important growth of online gaming. Today’s Massively Multiplayer Online Games (MMOGs) can contain millions of synchronous players spread across the world and participating with each other within a single shared game. The increasing number of players in Massively Multiplayer Online Games has led to some issues with the demand on server which generates a significant increase in costs for game industry and impacts the quality of service suffered by players. With the number of players gradually increasing, servers still need to work efficiently under heavy load and, new research is required to improve the established MMOG system architectures. In this paper, we survey load balancing techniques based on both client-server and peer-to-peer system to evaluate the current state of the art in MMOG scalability and to discuss the open issues. In addition, we proposed a new technique for dynamic load balancing based on hybrid peer-to-peer overlay architecture to deploy the players among the region, managing nodes joining and leaving operations, managing the nodes migration from a region to another, and managing the creation of the region in the virtual environment.

Keywords—MMOGs; P2P; client-server; load balancing; scalability;

I. INTRODUCTION

In recent years, there has been a new technical challenge emerging in the gaming industry which focused on the possibility of managing the resources of game servers for massively multiplayer online games (MMOGs). The major characteristic of massively multiplayer online games is supporting the large number of players, having hundreds of thousands or even millions of participants concurrently. Successful MMOG such as World of Warcraft with almost 12 million participations [1] have to provide a really scalable game world while preserving responsiveness.

MMOGs development faced many challenges. The most important challenge in MMOGs is scalability. Scalability indicates to the ability of system for dealing with rising number of players in the game world, without significant deterioration of the level of interactive experience. This is a critical problem which must be taken into consideration when designing MMOGs. Other requirements of online games are state consistency, responsiveness, reliability, security, and persistency [2].

The essential requirement of computer games is interactive and fast response for actions. When the player interacts with an object or other players in the game world, the player needs to respond immediately to the interaction. In client/server architecture, the server become overloaded and may become unable to react to the actions of the users. This problem affects the interactivity of the game and thus the gameplay experience of the users on the game. Using multi-server technique, the techniques introduced in different research work [3, 4, 5] for the purpose of addressing this problem and trying to achieve a high level of scalability. However, these methods incur extra costs because of the need for extra servers as well as the cost of their maintenance. To address this problem the work presented in [6, 7, 8], proposes to use peer-to-peer architecture to deploy the players among the region in the virtual world. In this paper, we will review these works and we discuss the limitations of existing load balancing techniques for MMOGs based on client/server and peer-to-peer architecture. In addition, we propose a new technique for load balancing for MMOGs based on hybrid structured peer-to-peer overlay architecture. The remainder of the paper is organised as follows. We introduce of MMOGs background in Section II. In section III, we discuss the existing techniques for load balancing Based on Client/server systems. Consecutively we present the existing architecture for load balancing based on peer-to-peer architecture in section IV. After the review of the state-of-the-art, we introduce a new technique for dynamic load balancing based on hybrid structured peer-to-peer overlay system in section V. Finally, section VI presented the conclusion of this work with suggestions for future work.

II. BACKGROUND

In this section, we present some of the fundamental concepts related to load balancing for MMOGs.

A. Massively Multiplayer Online Games (MMOGs)

Massively Multiplayer Online Games (MMOGs) are a genre of online game with thousands or even hundreds of thousands of participants who are playing simultaneously, and allow for the player to move and interact with each other in the virtual world. This gives the potential for a participant to socialize through online games, which gives a major advantage of MMOG over games with a single player. Nowadays, multiplayer online games become more popular in the game industry. This relate to the fact that the development of MMOG has three common sub-themes: Massively Multiplayer Online Role Playing Games (MMORPGs)such as World of Warcraft [1], Massively Multi-player Online Real Time Strategy (MMORTS) such as Prime World [9], and Massively Multiplayer Online First Person Shooter (MMOFPS) such as Planetside [10]. MMORPGs have become more common in the industry [11]. However, MMOFPS consist of having many
separated game services, with a handful of players each, but with minimal interactive environment capability [12]. The most substantial challenge in massively Multiplayer Online Games is scalability. This is referring to having the ability to cope with increasing number of players and allowing them to play together in the virtual world with a fair distribution of resources [3]. There are many MMOGs games such as World of Warcraft [1], EVE Online [13], and Final Fantasy XI [14] which have shown that MMOGs can be a prosperous business.

B. Client-Server Model

The client-server model has been the traditional method of writing networked multiplayer games, due to the simplicity and efficacy of the approach, with many examples of its usage. It consists of one or more authoritative game state servers, and one or more clients. The servers make changes to the game state that are propagated to the connected clients that in most cases (for cheat prevention and game fairness issues) cannot directly modify the state, but instead inform the server of their input for their particular player on the server. The client-server mechanism has been the traditional model for writing both simple networked multiplayer games and applications, and remains the simplest mechanism of consolidating the game state to ensure correct state synchronization.

For fast-paced action games, a client may run a local version of the game and synchronize it with the server game state to counteract latency between local player movement and remote player movement, and also to allow the server to provide fewer updates (otherwise, movement for other players which comes from the server may appear to stutter, as there is no interpolation). For slow-paced turn based games, the clients may receive entire states sequentially, or use state deltas to save bandwidth.

Regarding single server versus multi-server solutions, many classical MMOGs use a monolithic approach with a single machine or cluster of machines with an MMOG framework installed using code parallelisation techniques (usually either job-based or push-based with eventual consistency), with separate database servers in the same locality to periodically persist player data. An example of the task approach in use includes Eve Online [13], whose developers have stated that their system is monolithic with a very powerful master SQL server at the core.

Figure 1: MMOG Architecture

In Figure 1, we represent a common MMOG system architecture based on server cluster, involving game nodes, game database, chat server, billing server, web server, NPC control node, dispatcher and load balancer. This system architecture is centralized as all the clients communicate directly with one of the servers, which synchronizes the game state updates and communicates them to the clients. This is different from MMOG Peer-to-Peer in which clients can communicate directly among them. This is introduced in more details in the following section.

C. Peer-to-Peer Model

Peer-to-peer (P2P) system refers to a distributed system and every node in the network takes equal status and control. Recently, P2P networks have been growing faster and also become more widespread in the applications of the Internet. Peer-to-peer Networks can be divided into two types: pure P2P and hybrid P2P. A pure P2P architecture indicates to the network without a centralized server or control. However, hybrid P2P architecture refers to a network where the P2P system can be partially controlled via a centralized server. Peer-to-peer networks allow game designers to transfer a big part of the game processing and load of bandwidth to their participant’s peers. On the other words, the infrastructure of P2P authorizes to share the resources of a computer, such as CPU, bandwidth, and memory storage without necessity of centralized servers [15]. The main advantage of P2P architecture is better realized in file-sharing applications, however numerous attempts have been made to capitalize on this system architecture to applications other than file sharing, for instance, MMOG. P2P architecture has many established advantages, including scalability and reliability; cost sharing, resources aggregation, and autonomy. There are two main advantages very useful in solving some of the challenging issues in the traditional client-server based MMOG. The first is cost sharing. Instead of spending large sums of money on a single or clustered server, the P2P technique spread the cost among all the peers. But this must be supported by efficient load balancing techniques, which we introduce in the following section.

D. Load Balancing

Load balancing is a technique used to describe an endeavour to effectively distribute an application’s processing requirements across a number of servers. Load balancing technique relates to the capability to distribute the load of processing that occurred when the peers join and leave the system. Further, it is a major component in network systems to accomplish scalability. Load balancing describes the efficient process that describes the requirement of an application’s processing across a different number of servers and is the main concern for all distributed systems; in fact, the scalability problem in client/server MMOGs is in essence related to load balancing [16]. However, when applying load balancing across MMOGs infrastructure, it must be implemented in an effective way to prevent essential resources over-supplying on the server side. The basic method to load balancing was migrating nodes from heavily loaded to the other lightly loaded servers and thus redistributing the nodes load across distributed system. This is a relatively straightforward approach, but we have seen the limitation of this approach for
client/server architecture which requires adding more servers and resources. However, this approach is far from trivial in a peer-to-peer system. In P2P architecture, in which there are two main issues: firstly, how to determine if the node is overloaded or under-loaded, if there is no server; Secondly, how to find convenient partner for node with which to redistribute the load [17]. These are some of the issues for load balancing in pure P2P. We will discuss and propose some solutions based on hybrid P2P system architecture. Scalability issue in MMOG can be further improved by using communication reducing techniques, such as area of interest management (AOIM) which we introduce in the next section.

E. Interest Management

Interest management is a technique commonly used to determine the smallest amount of information that a peer requires, for the sake of presenting an accurate representation of the game world to players. In other words, interest management means communicating information on a need-to-know basis. This allows communicating the minimum amount of information that a peer needs to interact with other peers in the game world, in order to display an accurate state of the world to the players. This idea is not only related to P2P MMOGs, but has also been used in client/server architecture for virtual environment [18] and distributed environment [19].

Interest management schemes consist of two main categories: space-based and class-based [20]. Space-based refers to the interest management that is determined depending on the relative position of objects in the virtual environment, however class-based is determined based on the attributes of the object. Space-based interest management is usually depending on proximity, and could be comprehended in terms of an aura-nimbus information paradigm. The aura is defined as the area that borders the existence of an object in space, however, the nimbus or area-of-interest refers to the space in an object that can view other objects. Furthermore, the important aspect that related to interest management is Voronoi State Management (VSM) [21]. In the next section, we will review in more detail some recent load balancing technique based on client-server architecture.

III. LOAD BALANCING BASED ON CLIENT-SERVER SYSTEM

The section covers some existing load balancing techniques based on client-server. We also, discuss the limitations of these methods.

A. Triangle-based Obstacle-aware Load Balancing

In this approach, the authors utilize a hybrid technique that divides the main tasks the game’s logic has to perform. Firstly, interest management (IM) and secondly, state update dissemination. The main concept is partition game world into small triangles that take into account the world geometry such as walls. A cell consists of many triangles connected with each other within the virtual world. There are two factors considered in this technique: the load associated with performing game actions and the load incurred over interest management. When the server is overloaded it can either shed some triangles/tiles tile balancing from its cell [3].

The load balancing has achieved by two themes. The first idea is called Cell load model. In this model, when players and objects are equally distributed across the game world, the servers have the same cells size. However, it is uncommon that objects are equally distributed; also, most of players resort to the most interesting zones in the game world. Thus, servers holding the heavily populated tiles have to do more process for IM. The IM for each player has to be determined, and a large number of players in the cell. Therefore, the calculating load of the server based on two components: the number of players into server's cell and the number of objects and players the server has subscribed. The players are considered two times: once to perform IM and the other because they need to be considered for the IM calculation of other players in the game world. To deal with the load, giving the players inside the cell a high weight compares with the player outside the cell and object. Also, the authors define a threshold value to find out which node is overloaded ought to start to rid of tiles [3]. However, the limitation of this technique is not possible to achieved dynamic load balancing because assigned a threshold value for the node that prevent some node have a powerful resources to get more cell than others. As well as, the authors did not mention the obviously criteria to calculate the threshold value for each player. The second idea is called cell load distribution. When the cell skips threshold value and became overloaded, it tries to move some of its tiles to a neighbouring cell. It is quite important to select the tiles to be transferred. To do this, the cell calculates a priority value as follows; if the tile is neighbours and all members of the same cell, the priority is low such as 0. Otherwise, the priority is calculated according to the number of edge hops between this tile and the nearest tile of priority 0 [3].

B. A Fine Granularity Load Balancing

In this technique, the virtual world is divided into small regions based on kd-tree and achieves the load balancing of servers by repeatedly adjusting the split coordinates stored in its nodes. The important benefits for using kd-tree are: making this partitioning to allow a fine granularity to distribute the load and the readjustment of the regions becomes simpler. The load balancing approach is based on two main criteria: first, considering the servers as heterogeneous. It means each server may have a different quantity of resources. Second, the load of the servers are not related to the numbers of players, but to the amount of bandwidth required to send state update messages to them. Because the number of messages sent by the players to the server will be growing linearly with the increase of players number, while the number of state update messages sent by the server may not be good due to lack of bandwidth in the server. In this approach, using kd-tree with two dimensions, each node in the tree represents a region of the space and the node stored a split coordinate. Each node has two children and represents a subdivision of the region represented by the parent node. One of them represents the sub-region but have the parent node representing the region before the split coordinate. The region of the space is represented by the leaf node, which stores the list of avatars who are currently in that region. Ultimately, every leaf node is connected to a server of the game. When a server is
overloaded, it performs the load balancing by using the kd-tree to modify the split coordinates that define its region. In addition to reduce the amount of content managed by the server. Also, each node in the tree stores two values: capacity and load of the subtree. The calculation of the load and capacity of a non-leaf node are equal to sum of the load of its children. However, the leaf nodes have the same values of the server connected to each one of them [4]. The calculation of the load is depending on the way for distributing the players among the regions. The players are deployed between the servers according to the bandwidth for each server. However, this method is not optimal when the number of message transmitted between players is large. Using a brute-force method for calculating the loads by calculating the number of messages that should be obtained by each player by unit of time [4].

C. Load balancing for MMOGs

Lu et al., [5] presented a load balancing technique based on clustered server to achieve scalability. Allowing servers to transfer player actions to each other, however the responsibility for processing players’ actions remains with the server they are initially earmarked. The system consists of two main levels: the application level and the database level. Firstly, the player connects to the server cluster through a load balancer, and he/she is then linked to a particular server in the application level. Application level is dedicated to meet the runtime requirements of game play. Through the database level, an application server can have access to the virtual world constructs and players’ statistics via load balancer that exist between application level and the database level. Client-to-application and application-to-data load balancers are standard “off the shelf” network address translator (NAT) kind of load balancers [5]. But, this method is ineffective because the load balancer has not enough resources to cope with the increase of players’ number. Also, the cost of providing a large number of servers is very high as well as the cost of maintenance.

IV. Load Balancing Based on P2P System

In this section, we discuss the existing load balancing approaches based on P2P architecture.

A. Load balancing for non-uniform zones

Usually zone size in MMOGs is fixed, but there are some techniques that cope with dynamic region size using for instance Voronoi diagrams. Most of these techniques do not have load-handling mechanisms. The problem is the difficulty to predict player density or deploy the players before the start of a game. Thus, the server cannot share its resources with surrounding region servers. In this approach, [6] presented a load balancing mechanism where region shapes are not predefined. Consider a system with number of masters/servers with a large single region in the game world, the players can join and leave the game over time and at first only one server is involved. In this method, the game is divided dynamically depending on players’ logical position and interaction pattern. Consequently, the zone shapes are not uniform. The bisection algorithm is used to divide a region into two sub-region of roughly equal load while attempting to reduce the communication cost, such as the number of links crossing logical boundaries. Depending on the bisection methods, at the first time, the region is divided into one dimension to produce two sub-regions. Other partitions are made repeatedly in the new sub-regions if necessary[6]. However, this method is not efficient when the number of players is too large because we need extra server to manage the load, then the cost of provide server is high; furthermore, the maintenance of the servers is expensive.

B. Load balancing for uniform zones (Multilevel Multiphase)

Multilevel Multiphase Load Balancing (MMLB) approach is a method designed for fixed-size zones. This method was designed in two stages. The first stage works for the top level microcells-zones and cope with the inter-communication of server while organizing the load between the servers. However, the second stage works for the decomposition the top-level microcells into the deep-level microcells and sheds load with the help of zone masters. This technique helps the servers to limit structural reformation penalties also allow servers a provision to decrease the load in a step-by-step method. The authors considered that the best way to define the load is using message generation rate. Identifying the overloaded server was depending on assign a message threshold for all servers in the virtual work. The server is overloaded when the message rate larger than the threshold [6].

a) Load Balancing for Top-level Microcells

When there is overload in server, the load is relieved by migrating one or more microcells from the overloaded server to other lightly loaded servers. Because of the complexity of microcells movement, some intelligence is requested to determine the best potential set of microcells that can be moved. The major advantage of microcells movement from one master to another is to solve a hotspot problem [6], but this process will increased inter-server communication, as well as, the authors did not mentioned the way for selecting the best microcell which will be moved.

b) Load Handling through Deep-level Partitioning

When the extra-hotspot happens even after using top-level Microcells, the zone master uses the deep-level partitioning. In this approach, the zone is decomposed into seven hexagons. One zone is still completely inside the present zone but the other six small hexagons are connected with six neighbouring zones. When the system is overloaded, this method instantly involves all the enclosing zone masters. This method significantly decreases the load in the extra-hotspot region; however, this approach is unintelligent as, it is not necessary to involve and adversely impacts all the surrounding zone masters [6].

C. Structured Multi-Agent (SMA)

SMA consists of three layers: Pastry layer, logic layer and collaboration layer. Pastry layer is essentially used for resource locating. The manager in Pastry layer is responsible for finding resources and other nodes in easy way. However, logic layer is accountable for providing all the functions for normal game, and implemented as an interface for the logic
processing. Eventually, the collaboration layer is considered as a control centre for the management of all nodes and resources in SMA. In SMA, there are three kinds of nodes: data server, manager and agent. The responsibility of data server is for login authentication and states preservation. Manager node is allocated to an Area of Interest (AOI) for nodes organization. Subsequently, agent node is applied for processing events and act as a manager at the same time [7]. This module provides low scalability because all processing of the events is achieved by the agent only. Moreover, it cannot have a clear mechanism for resource management in an AOI.

a) Node joining and neighbour discovery in SMA

Firstly, the node joins in an AOI, by using the Pastry; the node should find the manager. Then it loads the whole information of agents from manager and obtains resources from them. If the manager of the zone is new, the resources will be received from data server. However, if the manager is set before, the resources will get from agents. After the relationships of nodes are confirmed, all the processing licenses of resources will be reset so that the events of a resource can be sent to the agent who has its processing license. However, the manager executes neighbour discovery if the nodes have not joined before. Otherwise, the node will obtain the information of neighbour agents immediately from the manager. The managers exchange agent lists when they establish the connections [7]. This method is efficient for joining and discovery of the neighbours but increase inter-server communication between the three layers. In the next section, we describe our dynamic load balancing technique for MMOGs based on hybrid structured P2P system.

V. DYNAMIC LOAD BALANCING BASED ON HYBRID STRUCTURED P2P OVERLAY SYSTEM

Massively Multiplayer Online Games are becoming more widespread each year with diverse new technology being released on platforms such as Steam [22]. These developments make the game more exciting to attract additional numbers of players to the game. The increase in the number of players creates issues to be faced by gaming companies at the present time because they do not have enough server or software to support the large number of players in the game world. Obviously, the best solution in this case is adding more servers. However, this solution is not permanently possible because of the cost of adding new server is high, and the communication of state dissemination is also more complex when the players are managed from different servers. To solve this problem, we propose a new technique to dynamically balance the load of players with high scalability, state consistency, and fault-tolerance. The benefit from load balancing is to ensure the players have the same opportunity to access the game. In our research, the load refers to the number of players associated to a single game server. In the next paragraph, we will describe the system in more details.

a) The architecture of MMOGs

In traditional online games, the server is the manager of all players and it stores all the game objects. This method is based on client-server architecture. It is unsuitable for a massive number of users in MMOGs. Although, the architecture of MMOGs based on pure P2P can improve the scalability, transferring the computational load and storage requirements from central server to peers in the network. This technique is not effective in this case as P2P architecture does not have efficient ways to control the behaviour of the players in the game world. To solve this problem, we propose to use hybrid P2P system [8] to control the game state and provide more security to the system. In our research, the architecture of MMOGs consists of centralized server, super-peer, clone-super-peer, and peers. Using hybrid P2P architecture to divide the virtual world into several manageable logical zones where each zone covers the players within this zone. The zones are managed by both super-peer and clone-super-peer, which connect to the server to access the peer's player’s information. The clone-super-peer can be either a super-peer in controlling another region or a peer in the current region. The peers in the region are connected to the super-peer and clone-super-peer using a mechanism such as Pastry [23] to reduce inter-communication.

b) Joining and leaving the game

The central server is responsible for registering the players when they join the game for the first time. Also, the server assigns a unique key to each player. This key will be used in message update dissemination, game state update, and deployment of the players among the regions. When the players join the game, the server allocates them to an available region to connect to. The players have many options to connect to the region who accept them. This process gives the player more flexibility to choose other participants to interact with them. Assigning the first player, who joins the game as a super-peer to the region. The super-peer work as a local server of region to manage message update dissemination and determine the area of interest (AOI) or so-called interest management (IM) for each player in that region. Also, the super-peer then sets a clone-super-peer. The clone-super-peer have two main benefits such as become a region super-peer after the super-peer has left the game or migrate to another region, and deal with the migration of players. When the normal peer wants to leave the game, they must inform the region’s super-peer. However, when the clone-super-peer wants to leave the game, they must also inform the region’s super-peer to set an alternative clone-super-peer for the region. Whereas, leaving of super-peer is done by informing both the server and the clone-super-peer and switching the clone-super-peer to become the region super-peer or to organise the election of the new super-peer for that region. Also, the new super-peer assigns a clone-super-peer to that region if none is currently assigned.

c) Players Deployment

The super-peer is calculating the resources such as CPU power, memory available, and bandwidth. This calculation is used to determine the number of peers that will be connected to it. The number of peers per region must be sent to the server to know that. Also, the current super-peer can evaluate the resources of each player connected to it to find the peer which has the most powerful resources to be the new super-peer node. This process is important to ensure the scalability of the system. Deployment of the peers will be the responsibility of
the centralized server. When the player joins for first time, the server will give him a ticket with available region that can connect to them. This process is important to ensure both the organizational and the performance of the game world.

a) Dynamic load balancing

As mentioned before, the game server knows about the possible number of players which connect to each region. When the player joins the game, the server gives him/her a ticket contains the list of available region that it is possible to connect. The number of players per region is depending on the powerful of resources of super-peer. The players will be distributed among the region without, however, if the players interested in playing in certain region, and the region becomes overloaded, the current region will be split into two region, as well as assign the clone-super-peer as a super-peer to the new region. Figure 2 depicts the proposed hybrid architecture assuming the clone-super-peer in one region is also super-peer in another region.

![Figure 2: The proposed architecture of MMOGs](image-url)

VI. CONCLUSION AND FUTURE WORK

This paper presented load balancing techniques for MMOGs based on both client/server and peer-to-peer architecture. The advantages, disadvantages and restrictions of each method have been discussed and with the idea of proposing an approach that solves these limitations for MMOGs load balancing. A new technique for dynamic load balancing based on hybrid structured P2P overlay system has been proposed to ensure high scalability, state consistency, fault-tolerance, high performance and organization. The proposed technique has also the capability of allowing players to migrate from one region to another and participate with other player in other regions.

This work can be extended into two different areas. The first theme will be on simulating the MMOGs infrastructure to test the resources of the peers such as the power of the CPU, the bandwidth, and memory availability, and ensure which resource may has an effect on the scalability, consistency, latency, and fault-tolerance. The second theme will be the design and development of MMOGs framework. The idea is to design an instance of real-world online MMOGs based on hybrid structured P2P overlay architecture to carry out more realistic evaluation and analysis.

VII. REFERENCES