SEIKEI UNIVERSITY

Dynamic Data Allocation Method for Web-based Multiserver Systems

DISSERTATION

submitted in satisfaction of the requirements
for the degree of

DOCTOR OF SCIENCE AND TECHNOLOGY

in Computer and Information Science
by

Masaki Kohana

Dissertation committee:
Professor Shusuke Okamoto, Committee Chair
Professor Makoto Takizawa
Professor Atsuko Ikegami
2012

Contents

List of Figures

List of Tables

Acknowledgment

The author would like to gratitude to many people who support the completing the degree.

First of all, the author is deeply grateful to his supervisor, Professor Shusuke Okamoto at Seikei University, Department of Information and Sciences, Computer System Laboratory, who provided generous support and suggestions. The author is always helped by his comment and suggestion.

The author would also like to express his gratitude to Professor Makoto Takizawa and Professor Atsuko Ikegami at Seikei University, Department of Information and Science. Their opinions and information have helped the author throughout the production of this study.

Professor Tatsuhiro Yonekura and Professor Masaru Kamada at Ibaraki University, Faculty of Engineering, gave the author meticulous comments and discussions.

The author would like to thank the members of Computer System Laboratory at Seikei University. They gave the author lively discussion.

Lastly, the author's deepest appreciation goes to his parents and his friends.

1 Introduction

1.1 Web-based Application

Web-based application is a popular type of software and takes a form of server-client software. Web-based application runs on a web server and displays the result on a client. In the early days of Web, web server delivered only static HTML file to client. By the appearance of CGI, web server could create HTML documents dynamically according to an input data from a user. Thereby, various web-based applications could be developed. The using client-side scripting language such as JavaScript provides dynamic user-interface on a web browser. And Ajax provides asynchronous communication between server and client. Then, the more flexible web-based application could be developed. Due to the speeding up JavaScript on web browsers, web browser could process the more complex computing.

Web-based application uses web browser as a client software. At the server-side, CGI script is used to handle requests from web browsers. CGI script is typically written by php, perl and ruby. Web browser executes browser-supported programming language such as JavaScript and communicates with a web server. It can be available if there are a web browser and the Internet connection. And it is independent from Operating System (OS) and computer architecture. Thus, the user does not need any special client software and any special settings of computer. It is preferable for the user who does not have the expertise of the computer. Furthermore, the user data is stored into database on the database server. Then, the user can access to own data from various computers and various places.

Additionally, Ultra Mobile PCs (UMPCs) and smartphone such as iPhone or Android phone becomes popular and FREESPOT is increasing. Hereby, the number of web-based applications has been increasing. In fact, there are some popular web-based applications such as Facebook, Gmail, Twitter and so on.

In the future, the number of users for web-based applications might continuously

increases. Thus, techniques to serve a large number of users are needed.

1.2 Multi-player Online Game

Multi-player Online Game (MORPG) is one of the most popular online game. Players create their own avatar in a virtual game world. The player operates own avatar using mouse and arrow key and interacts with the other players using chat message and character gesture. MORPG which has a large number of players is called Massively Multi-player Online Game (MMORPG). There are a large number of popular MMORPGs such as Final Fantasy 14, Dragon Quest 10, World of War Craft and so on.

The most MMORPGs require a special client software and they are provided for Windows OS. On the other hand, there are some MORPGs which are provided as a web-based application such as ELEMENTALIA, Fragoria and Rune Scape. This type of MORPG does not require any special software and independents from OS and architecture.

Most online games divide a virtual game world into some regions. Each region is managed by a separate server. Thus, when the avatar across the region, the client software connects to a new server and reload the virtual game world. In addition, some online games limits the number of avatars in a region. This reason is to control the server load.

1.3 Motivation

Web-based application has some advantages. However, there are some disadvantages. The access congestion to a web server causes database access conflict in case of higher frequent communication. And the different access latency leads to unfair situation. Short latency and high fairness are important for this kind of online game. Furthermore, a player might converge on certain region in the virtual game world. It leads

to higher server load. The game provider must be required huge investment to handle the peak demand of players.

This research focuses on web-based MORPG which has more frequent request to a web server and introduces a load-distribution system using multiple web servers to resolve the problem. The entire game world is divided into small blocks and each ownership of block is assigned to a web server. A web browser that is a client software connects to a web server. Thus, the access congestion is resolved. However, when the block data does not available locally, the server that receives a request should retrieve the data from the other server that has the data. This communication among servers also becomes an overhead.

In order to reduce the frequency of the communication, this research proposes a dynamic data reallocation method. This method attempts a data allocation that has the least frequent communication. Two types of data reallocation methods are proposed in this research. The one is the data reallocation based on local data. The another is the data reallocation based on global data.

With the allocation based on local data, each server determines which server should manage the block independently. Three rules are introduced to determine when a block moves.

With the allocation based on global data, the system collects the entire game information from all servers. To obtain an optimal data allocation, the block assignment is formulated as a combinational optimization problem. To understand an optimal data allocation, this formulation is solved by using an exact algorithm. Furthermore, it is solved by using a meta-heuristic algorithm to obtain an optimal allocation quickly.

1.4 Organization of this paper

This paper is organized as follows. Literature Survey is discussed in Section 2. It shows some virtual environment system and dynamic load distribution system for

online games. Section 3 shows the system architecture of this research, which includes the overview of the game setting and describes the communication among servers called ‘remote access‘ and the dynamic data reallocation method called ‘moving home.‘ Section 4 proposes the data reallocation method based on local data. Using this method, each server determines which server should manage the data independently of each other. The data reallocation method based on global data is described in section 5. This method uses the data from all servers to determine which server should manage the data. In order to achieve an optimal data allocation using this method, the data allocation problem is solved as a combinational optimization problem. This problem is solved by an exact algorithm using IBM CPLEX and a tabu search algorithm which is a meta-heuristic algorithm. Finally, this paper is concluded in section 6.

2 Literature Survey

This section describes the literature relevant to this paper. There are numerous virtual world services, including Second Life, World of Warcraft, and Meet-Me [27][28][29]. The main topic of this paper is the load distribution on servers in a multi-server web-based MORPG system. We aim to make load balanced by using a data allocation technique. There is some existing research relevant to this work.

2.1 Analysis of virtual environment

Daniel Pittman and Chris GauthierDickey measured the virtual population in Massively Multiplayer Online Games (MMOGs) and investigated the player movement and distribution[1]. To measure the population on virtual world, they designed an add-on for World of Warcraft which is a most popular MMOG. They cleared that the peak population appears at the evening of the day and week end. In addition, they also cleared that the users converge on a few areas such as the large city and they visit a few areas during their game playing session. This means that some servers which manages crowded area has higher server load. Thus, the load-distribution method to handle the higher server load is needed.

Wu-chang Feng et al. provided a comprehensive analysis of popular online multiplayer game servers[6]. They stated that the workload is characterized by highly periodic bursts of small packets, with predictable long-term rates. Additionally, they showed that game players themselves have interesting session-times and geographic locations.

Waldo discussed the necessity of finding and constructing environments for online games and virtual worlds[7]. He stated that problem solving in this new environment was a great challenge, and that a new approach to software development was required for multi-threaded multi-core distributed systems.

Kuan-Ta Chen et al. stated that system-level performance such as bandwidth

and latency is unlikely to influence how satisfactory a user finds a system to be[5]. Nevertheless, a user's perception is inevitably influenced by the application design and implementation. They showed that game play time is strongly related to network QoS; thus, network QoS is a potential indicator of user satisfaction.

Frank Glinka et al. provided a comprehensive analysis of multi-server distribution mechanisms, zoning, instancing, and replication[11]. Their approach is efficiently supported by a current real-time framework implementation, which provides both a high level of abstraction and preserves design flexibility in single- and multi-server game engines.

2.2 Implementation of virtual environment

Ahmed Abdelkhalek and Angelos Bilas investigated the parallelization and the scalability of interactive multiplayer game servers [2]. They proposed the threaded game server. Their main challenges were the task decomposition and the synchronization for correct game processing. They parallelize Quake game server using multi-threads. It uses a shared memory architecture. In the system, the locking synchronization time grows from 2 % to 35 % of the server execution time. However, using locking method based on application specific knowledge could reduce the locking time about 20 % of total execution time. Their threaded game server can support 25 % more players than the sequential game server.

To maintain consistency of online game, Dead-Reckoning (DR) is an important method. It predicts the player movement. It could filter most unnecessary state updates. Yi Zhang et al. proposed a method named Globally Synchronized Dead-Reckoning with Local Lag, which combined local lag and Globally Synchronized Dead-Reckoning (GS-DR) [4]. This method can eliminate the after-inconsistency and decrease the before-inconsistency. This method showed that by combining local lag and GS-DR, the constraint on selecting lag value is removed and a lag, which is smaller

than typical network transmission delay, could be used. As a result, the system which has large network transmission delay uses the application of local lag.

2.3 Virtual environment Architecture

Jeremy Burn et al. focused on the network topology [3]. They introduced the notion of critical response time in order to optimize playability and fairness. They formulated the optimal critical response time for the server selection problem and solved it for the small networks of three different types of network topology. They also introduced an approximate heuristic solution for large networks. The heuristic approach closed to an optimal critical response time.

Jared Jardine et al. developed a hybrid game architecture. This architecture uses a combination of client-server and peer-to-peer architectures[10]. This combination ensures that a sufficient number of players are capable of functioning as regional servers, thus keeping the latency low, and allowing the game to handle a great number of moves per second.

Shun-Yun Hu et al. proposed Voronoi State Management (VSM) to maintain object states for peer-to-peer-based virtual worlds[12]. VSM supports existing consistency control to enable scalable, load-balanced, and fault-tolerant virtual environment (VE) state management. VSM integrates both client-server and peer-to-peer VE designs in a unified approach.

2.4 Load balancing approach

Radu Prodan et al. proposed a new prediction-based method for dynamic resource provisioning and scaling in massively multi-player online games (MMOGs) in a distributed grid environment[8][9]. Their method delivers computing resource to server dynamically and manages the resource in data center. They specified the various type of player interaction and player population trend. This method estimates the MMOG

resource demand dynamically and provides computing resource to game server according to the entity distribution in the game world. They tunes a neural network-based predictor to achieve good accuracy consistency. This dynamic resource provisioning method was more efficient than the static provisioning. Their method reduces the average over-allocation from 250% to around 25%.

The dynamic resource provisioning is also used to Database-as-a-Service (DBaaS). Jie Zhu et. al. proposed a dynamic resource allocation framework for DBaaS[24]. To maximize resource utilization, their framework reallocates resources to tenants. But the service level agreement (SLA) should be satisfied. Thus, they models the resource reallocation problem as a modified unbounded knapsack problem. It introduces an additional constraint. Their experimental result shows that the framework always achieves higher resource utilization.

Fengyun et al. proposed a technique to balance the load over the server cluster for MMORPG[18]. It is simple but effective, while maintains the flexibility of a cluster. They takes a behavioural approach to balance load. And it distributes the player connection over the clustered server in a round-robin fashion. It afforded the higher degree of interactivity. Their approach is scalable while maintaining real-time requirements.

Carlos et al. proposed a load balancing scheme for distributed MMOG servers[19]. They has two goals, allocating load on server nodes proportionally and reducing the inter-server communication overhead. They considered the upload bandwidth occupation of the server as the load. And they used a greedy graph partition growing algorithm to reduce the inter-server communication overhead.

Mejdi Eraslan et al. proposed a transport system for a distributed virtual environment with IPv6[20]. They showed the limitation of the IPv4 and proposed a network architecture using the IPv6 to support the per-packet level QoS for the large-scale virtual environment which needs the great bandwidth.

Herbert Jordan et al. proposed a game load management technique divided into global and local layer[21]. This technique is divided into two layers (global and local).

The global layer uses peer to peer architecture to assign the responsibility of managing game region to a data center. The local layer is done within individual data center. It maintains the server instance for the assigned obligations. To maximize the resource utilization, they uses two meta-heuristics based on a bin-packing problem. Their approach achieved the cost reduction in maintaining MMOG session by up to 60% while maintaining QoS in 99%.

Huang et al. proposed a neighboring-based distributed dynamic load balance policy for achieving the load balancing in P2P grid system[22]. Their approach migrates the job to the neighboring site with minimal job turnaround time. Their experimental result shows the approach improves performance.

These approaches use some middleware to achieve efficient resource usage. As mentioned earlier, the approach in this papaer targets web-based applications. It aims to achieve load balancing without modifying the web server. Thus, a dynamic data reallocation method is chosen to balance the server load, with which the approach aims to achieve enhanced player capability.

2.5 Division of virtual environment

There are a number of methods that use multiple servers and divide the virtual world into areas. De Vleeshauwer et al. proposed dynamic microcell assignment [13][14]. In this approach, the virtual world is divided into small cells, called microcells. These microcells are then dynamically assigned to a set of servers to address the high, dynamically varying player density. This architecture is important for managing a large number of users. Their algorithm achieved a load reduction of up to 30%.

Roman et al. investigated an architecture of a unified virtual world and proposed a load-balancing method[23]. In this architecture, the virtual game world is not divided. But some server divides the client processing with the rectangle region. This system changes the size of region according to the client movement. These regions might

overlap with the others. Their proposed load balancer has two modes. The first mode attempts to balance the load. The second mode resolves any overlap inefficiencies.

2.6 Web-based virtual environments

There are some web-based virtual environments, RuneScape, Fragoria, and ELEMENTALIA[30][31][32]. In addition, some researchers have proposed applications for web-based virtual environments. Hayer et al. proposed a tele-operated laboratory [15]. They designed a web-based virtual environment that they used to control laboratory experiments. Champion conducted an online exploration of archaeological reconstructions [16]. This exploration employed JavaScript and an XML-based application.

2.7 Live migration of virtual machine

The data allocation method is looks like a live migration of virtual machines which is a key technique for cloud computing. Yi Zhao et al. proposed an implementation with an adaptive live migration of virtual machines[25]. It uses shared storage to decrease the migration time. They introduced a COMPARE_AND_BALANCE that is a distributed algorithm.

Kejiang Ye et al. focuses on the live migration mechanism[26]. In the mechanism, the multiple virtual machines has different resource reservation method. They evaluates the performance impacts of their migration framework for both a source machine and a target machine. Furthermore, they analyzes the efficiency of parallel migration strategy. Their experimental result cleared that the live migration of virtual machine brings some performance overhead.

2.8 Conclusion of literature survey

The most of introduced research in this section employ the computing resource allocation or job migration in distributed system. And some research used middlewares to achieve the objective.

This paper aim to achieve load balancing without modifying web servers. And the data required by players differ for every player. Therefore, the approach in this paper uses a dynamic data reallocation method to balance the server load and aim to achieve enhanced player capability. This dynamic data reallocation method is performed on CGI and data transfer are performed on the HTTP protocol. Thus, the modifying web server is not needed.

We adopt a latency threshold to evaluate the approach. Claypool et al. discussed the precision and latency required by different categories of online games[17]. They proposed that the latency required by a MORPG should be 500 ms. Based on this proposal, we aimed at adjusting the latency of our rules to 500 ms or less.

3 System Architecture

This section describes an overview of the web-based MORPG system, which includes the server construction and the virtual game world construction. And this section also describes a communication among web servers and a dynamic data reallocation method.

3.1 System Overview

Figure 1 shows a login page of the MORPG in this research. A player enters the name of the own avatar and password, and chooses a picture of the avatar. After that, the player pushes the start button. Then, the page transfer to the main game screen.

Figure 2 is a game screen of the MORPG. There are two kinds of characters. Each character name is displayed at the underfoot. The elephant and the frog are avatars of the own players. Each avatar is operated by the own player using mouse and arrow key. Each player explores a game world and interacts with the other characters using the own avatar. The round shaped characters are non-player characters (NPCs). They are operated by a program on a server.

The text box at the upper side is a chat form. To send a message to the other players, the player inputs a message to the chat form and press the send button. The player can communicate with the others by their message.

Figure 3 shows the relationship between the game world and the view area on a web browser. The entire game world is divided into small blocks. Each ownership of block is allocated to a web server. A block contains game data for characters who stays there. The server which has the ownership of the block is called home of the block and has a responsible to update data and to provide the data to clients and the other servers. The rectangle with thick boarder indicates a view area on a web browser of a player. A web browser locates the own avatar at the center of view area and displays the game world included in the view area. Thus, the web browser retrieves the game

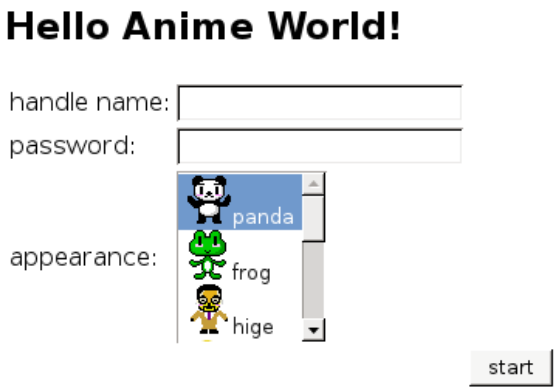


Figure 1: Login Page

data included in the view area.

3.2 Server Structure

Figure 4 shows the server structure. There are six web browsers and four web servers. Each web browser connects to only a web server. All web servers are mutually connected to all the other web servers to share the data. Each web server has two key CGIs, getinst.cgi and getorg.cgi, and two databases, character.db and home.db. Getinst.cgi is used to handle the request from a web browser. Getorg.cgi is used to handle the request from the other web server. Character.db stores the characters data on the block that is managed by the server. Home.db stores which server is home of the block.

When the block data that required from the web browser does not available locally, the server that received a request should invokes the getorg.cgi on the other web server that is the home of the block to retrieve the block data. The invoking getorg.cgi is called remote access.

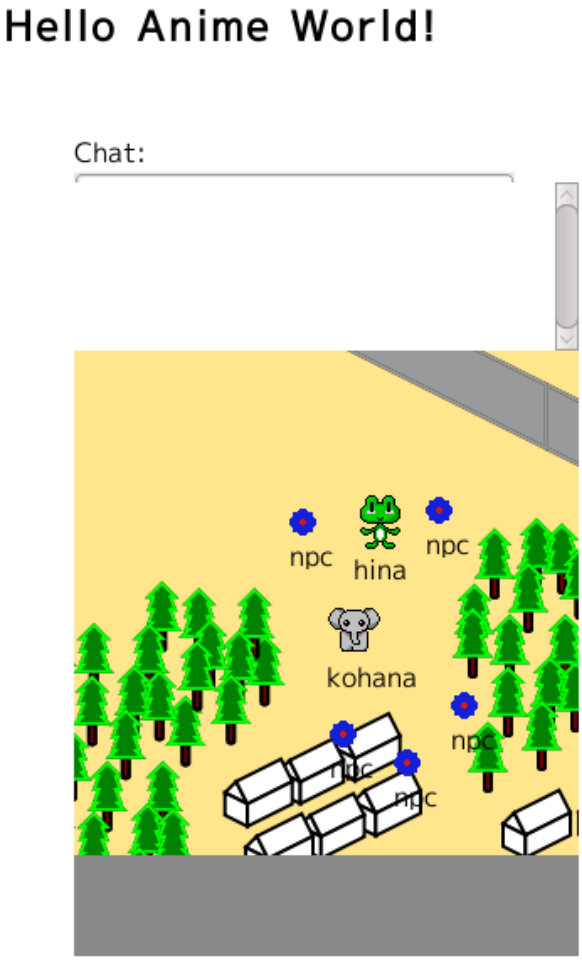


Figure 2: Game Screen

3.3 Remote Access

Figure 5 illustrates a typical case of remote access. There are two web browsers and two web servers. The server A is home of white blocks and the server B is home of gray blocks. The web browser that operates the elephant connects to server A and the browser that operates the frog connects to server B.

The web browser for the elephant sends a request to server A. Then, server A retrieves the block data included in the view area of the elephant. However, the view area of the elephant includes two gray block. Therefore, server A invokes the getorg.cgi

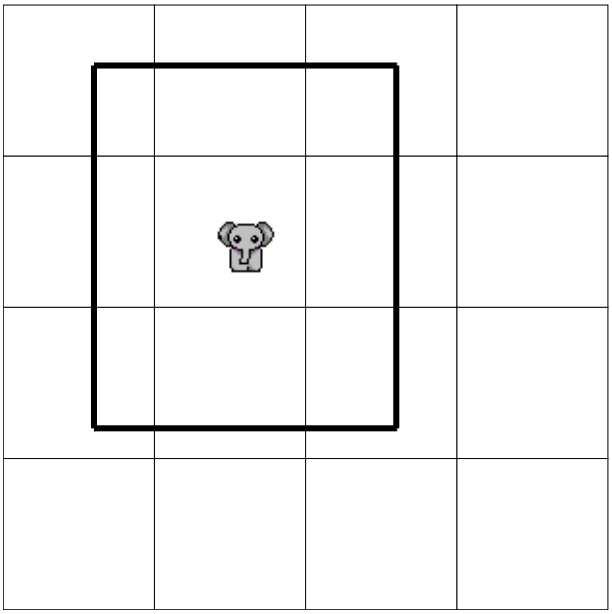


Figure 3: Game world and View area

on server B that is home of gray blocks. After retrieving the data of gray blocks, server A replies the required data to the web browser. On the other hand, the web browser for the frog sends a request to server B. Then, the view area of the frog includes only gray blocks. Therefore, server B does not need any remote access.

3.4 Moving Home

Using multiple web servers, the access congestion to a web server could resolved. However, making a remote access becomes an overhead. When a web browser requires a remote access, a web server performs as a client and sends a request to a home of block. The communication time doubles. Thus, the communication time becomes longer depending on the number of times remote access, which increases the latency. Increasing the frequency of remote access makes higher overhead and increases the server load. Thus, the latency of server becomes long and playability gets worse. The frequency of remote access changes according to the block allocation and the avatar location.

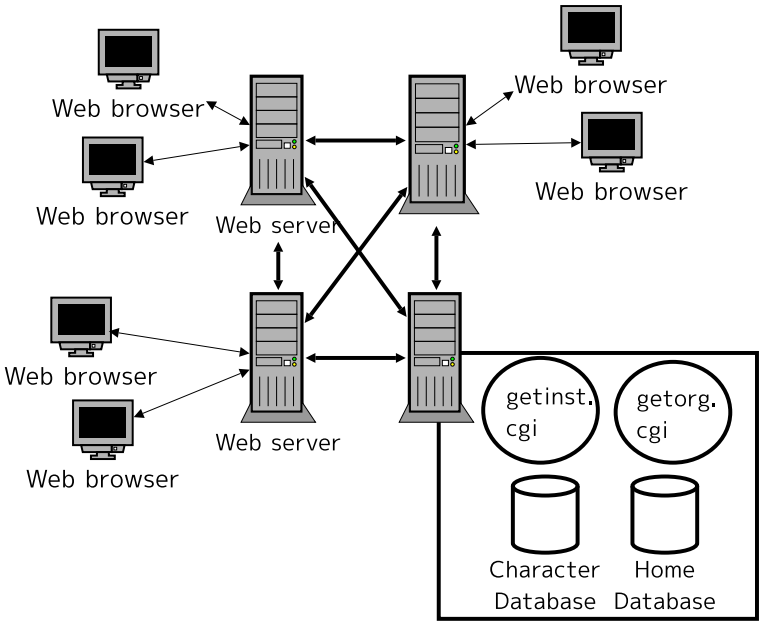


Figure 4: Server structure

The avatar location changes every moment. Therefore, the optimal block allocation is needed to achieve the high playability. The system in this research changes the block allocation in progress of MORPG to reduce the frequency of remote access. This change of block allocation is called moving home.

Figure 6 shows an example of moving home. There are two web servers and a web browser. The server A is home of white blocks and the server B is home of gray blocks. The web browser for the elephant connects to server A. The browser sends a request to server A. Server A invokes the getorg.cgi on server B because the view area of the elephant includes gray blocks. However, the other avatars does not need there gray blocks. Then, the gray blocks included in the view area of the elephant should be managed by server A. After moving the home of these gray blocks from server B to server A, any remote accesses become unnecessary for server A.

As an example of above, the frequency of remote access could be reduced by moving home. This research introduces two types of implementations for moving home. The

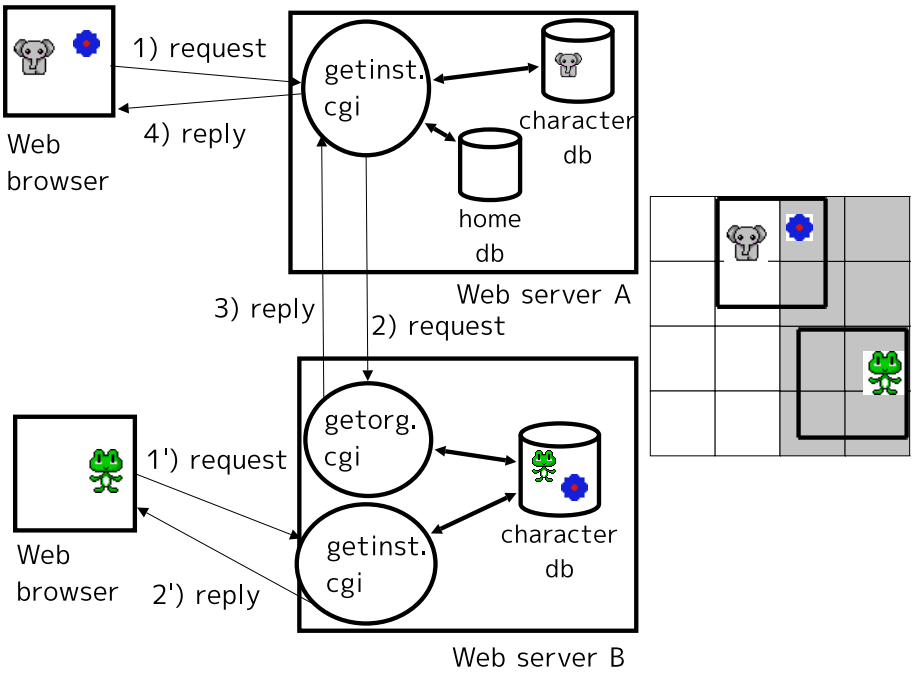


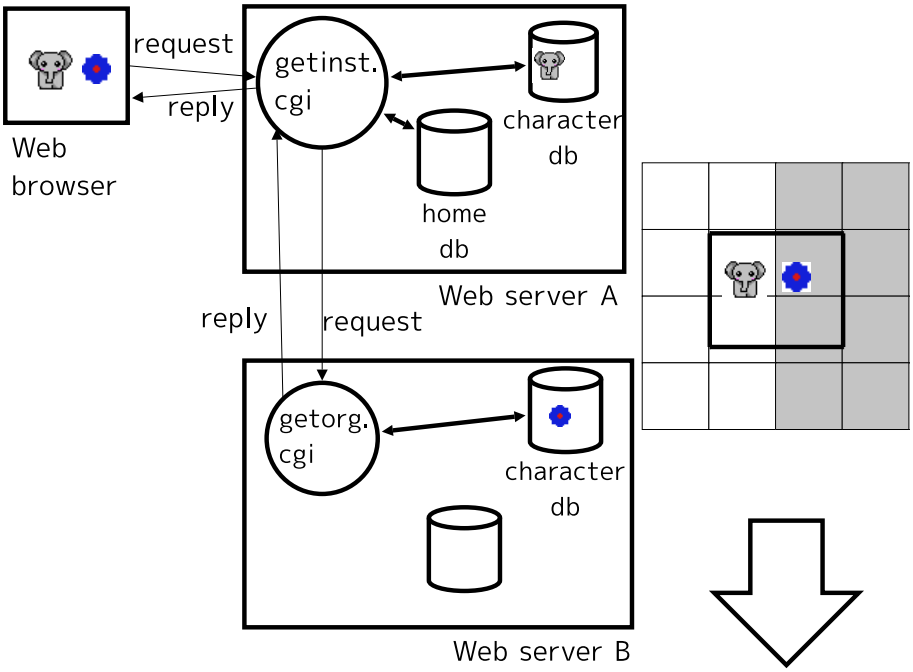
Figure 5: A Case of Remote Access

one is based on local data and section 4 describes the detail of it. The another one is based on global data. It is described at section 5.

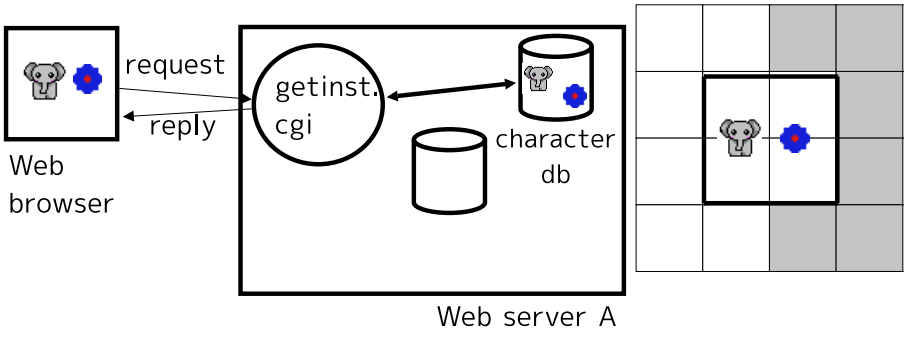
3.5 Software Implementation

3.5.1 Database Setting

There are two databases in this system, character.db and home.db. Character.db stores the information of characters. This database has two kinds of tables (avatar, NPC). Table 1 and 2 shows the table of avatar and NPC respectively. Each table contains ID, block number, location, picture file name, size of pictures and names. The x and y indicate a location of a character. The width and the height indicate the size of the picture. Additionally, the avatar table contains the password and the login state of player. In contrast, the NPC table contains the name of the function. It indicates the



a) before move



b) after move

Figure 6: A Case of Moving Home

id	block	x	y	pic	width	height	name	passwd	login
1	36	643	263	panda0.png	32	32	test0	0000	logout
2	96	623	630	elephant0.png	32	32	test1	0000	login
3	47	279	331	frog0.png	32	32	test2	0000	login
4	79	445	522	domo0.png	32	32	test3	0000	logout

Table 1: Table of Avatar

id	block	x	y	width	height	pic	name	func
1	133	1383	886	16	16	iasl2_xpm.png	npc0	trans_npc_s4
2	22	777	196	16	16	iasl2_xpm.png	npc1	trans_npc_s2
3	207	1293	1335	16	16	iasl2_xpm.png	npc2	trans_npc_3
4	73	1386	492	16	16	iasl2_xpm.png	npc3	trans_npc_s3

Table 2: Table of NPC

next behavior of the NPC.

Home.db stores information about which web server is a home for a block. Table 3 shows the contents of the home.db. An id field indicates a block number and a host indicates an IP address of a web server.

3.5.2 Login.cgi

Login.cgi is invoked from the login page. It receives an avatar name, password and a picture of the avatar. The received message is HTTP/1.1 POST method form. Figure 7 shows an example of this message.

When a player logs in at the first time, the login.cgi locates the avatar at random position in the game world. After that, it checks the home of the block that the avatar is in. If the home of the block is not local server, the login.cgi should invokes the other login.cgi on the home of the block. After that, the login.cgi updates the login status of the avatar and returns the avatar position to the web browser.

id	host
0	192.168.114.200
1	192.168.114.200
2	192.168.114.201
3	192.168.114.201
4	192.168.114.202
5	192.168.114.203

Table 3: Home.db

```
POST /cgi-bin/rpgserver/login.cgi HTTP/1.1
Host:192.168.114.70
Connection:close
Content-Length:32

nm=kohana&ps=0123&pic=panda0.png
```

Figure 7: Example of POST message

3.5.3 Getinst.cgi

Figure 8 is the process flow of getinst.cgi. This CGI receives a request from a web browser. Then, it checks which blocks are included in the view area of the web browser. After that, this CGI checks the home of blocks that included in the view area. This CGI updates the NPCs information on the local block. When the home of some blocks is the other web server, this CGI invokes the getorg.cgi on the server that is the home of the block. If the home of the block already moves to the other server, the getinst.cgi receives the new home of the block and invokes the getorg.cgi on the new home of the block. The getinst.cgi repeats this invoking until the request achieves to the current home of the block. When the moving home occurs, this CGI receives the avatars and NPCs data on the block. And it stores these data to the local database, and updates

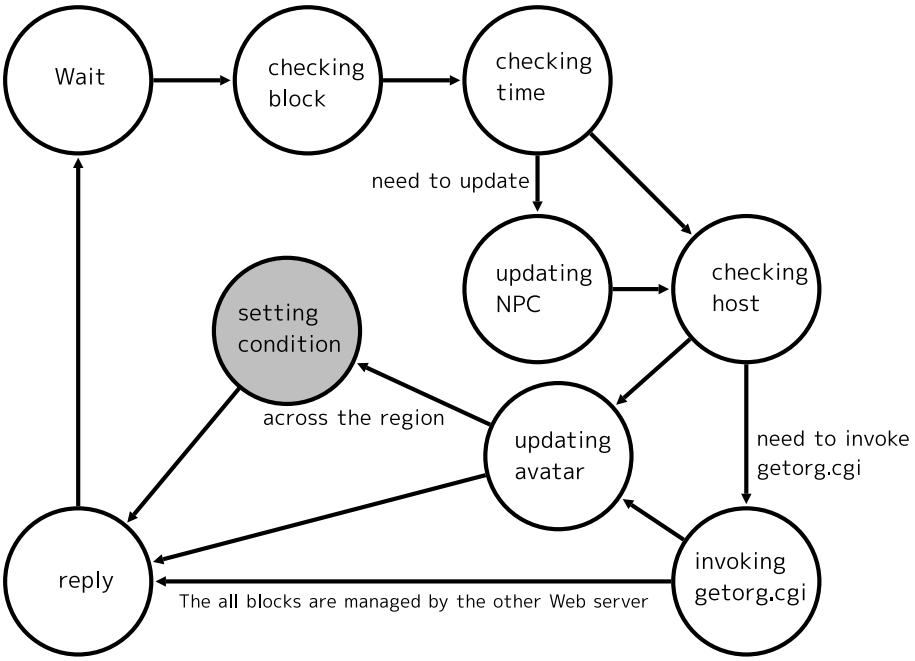


Figure 8: Process flow of Getinst.cgi

the home information in the home.db. If the avatar that managed by the web browser exists in the local block, this CGI updates the avatar information. Otherwise, the avatar information is updated by the getorg.cgi on the home of the block. After the updating the avatar, getinst.cgi updates the condition of the moving home. The detail of the condition is described in the following section. This CGI retrieves the character information on the local block from the character.db. Finally, the getinst.cgi replies the data to the web browser.

3.5.4 Getorg.cgi

Figure 9 is the process flow of getorg.cgi. This CGI is used to share the block data among web servers. Getorg.cgi receives a request from getinst.cgi on the other web server. If the home of the block already moves to the other server, getorg.cgi replies the new home of the block to the invoker. When the current status satisfies the condition

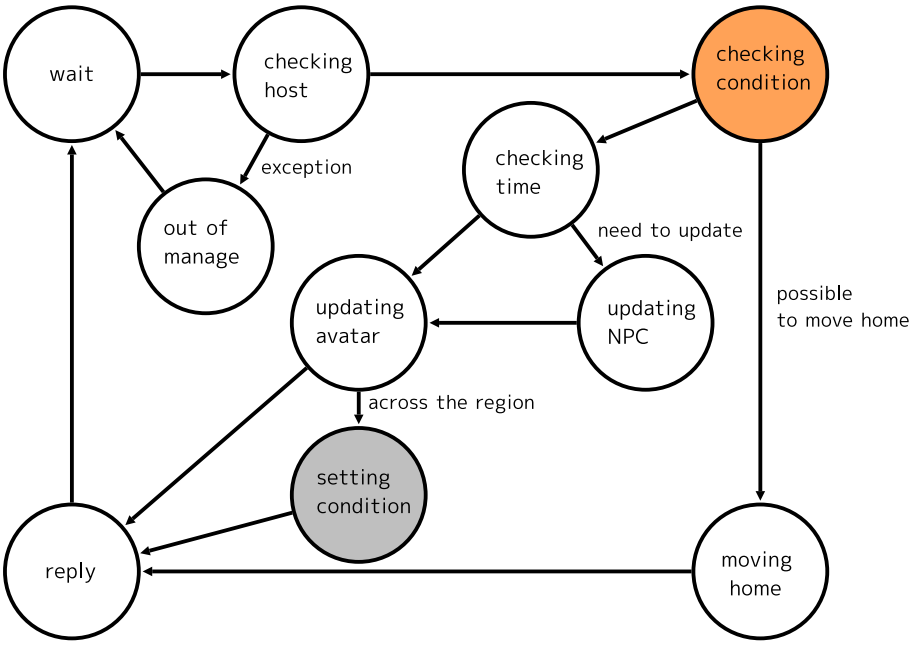


Figure 9: Process flow of Getorg.cgi

of the moving home, getorg.cgi retrieves the data of all characters that includes the logged out avatars and sends these data and the block number to the invoker. After that, the getorg.cgi deletes the character data that in the block from the character.db and sets the new home of the block to the home.db. When the moving home does not occur, like getinst.cgi, this CGI updates the avatar and the NPCs information and also updates the condition of the moving home. And it retrieves the required information from the character.db and replies the information to the invoker.

3.5.5 Communication Protocol

There are two kinds of communication in this system. The one is the communication between web browser and web server. The another is the communication among web servers. Both types of communication conform to HTTP/1.1 and POST method. A request includes an ID of character, a character name, the location, the range of view area and block number.

```
({'show':[
['npc46',1932,60,'iasl2_xpm.png',16,16,1,'npc'],
['npc29',1862,170,'iasl2_xpm.png',16,16,1,'npc'
]]})
```

Figure 10: JSON for characters

Web server replies the information on the JSON format. Figure 10 shows an example of the JSON format.

This message includes the two information of NPC. Each information contains the ID, the location, the picture file name, the size of the picture, the clickable flag and the name of NPC. The web browser displays the characters according to this message.

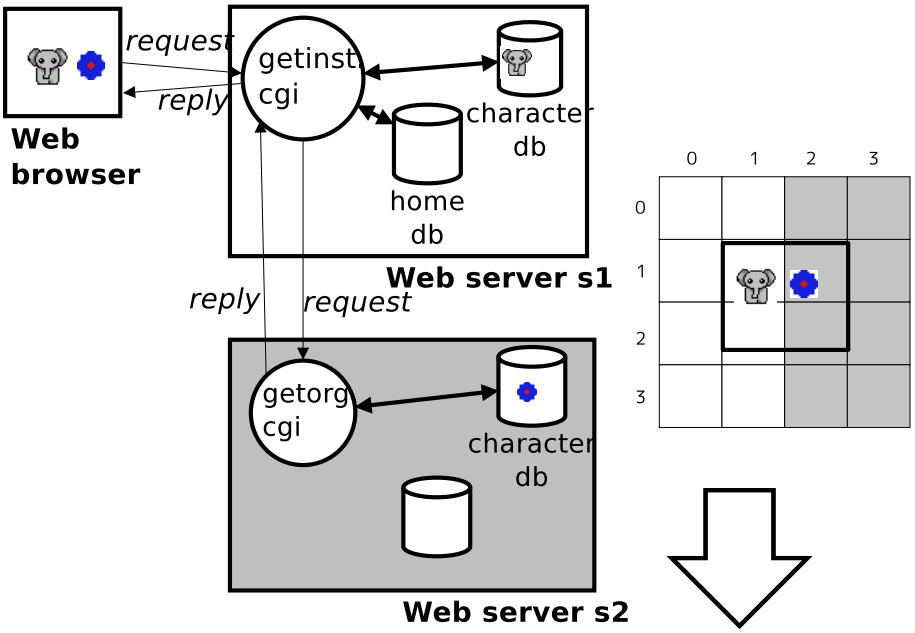
4 Home allocation based on local data

This section describes the block allocation based on local data. Using this allocation method, each server determines which server should manage the block independently of each other. In this way, each server uses only local data. Thus, the sharing data among servers is not needed. This research proposes three types of rules for the moving home based on local data to help the decision making, LRC rule, time-based rule and count-based rule.

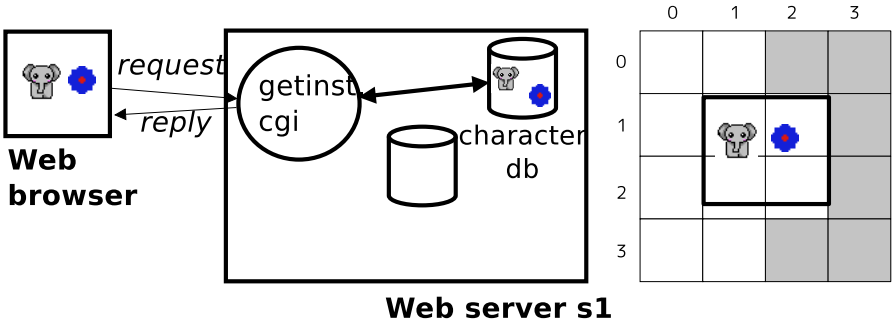
4.1 LRC Rule

LRC rule is based on how many avatars are associated with the current home block. This rule uses local-reference counter (LRC) to count the number of avatars that require the block. Each block has a LRC. Each block is associated with its LRC. Home of a block is moved from local to remote when a remote server sends a request for the block and LRC value of this block is one, in other words, in case where only an avatar on the remote server requires the block.

Figure 11 shows an example of how to use LRC rule. There are two web servers, s1 and s2. Server s1 is home of white blocks and server s2 is home of gray blocks. The web browser for the elephant connects to server s1. The view area of the elephant includes four blocks from (1, 1) to (2, 2). The elephant avatar is in block (1, 1), currently home on s1. The NPC is in block (2, 1), currently home of s2. Server s1 invokes the getorg.cgi on server s2 because the view area of the elephant includes gray blocks (2, 1) and (2, 2). However, no other avatars require the blocks (2, 1) and (2, 2). This is indicated by the LRC values for block (2, 1) and (2, 2), both of which are 1. As a result, server s1 becomes the home for (2, 1) and (2, 2), because server s2 does not need to manage these blocks, and there is no longer a need for remote access.



a) before move



b) after move

Figure 11: LRC rule

id	host	start	end
0	192.168.114.70	1350636437	1350636496
0	192.168.114.71	1350636448	1350636514
0	192.168.114.72	0	0
0	192.168.114.73	0	0
1	192.168.114.70	1350636027	1350636620
1	192.168.114.71	1350636102	1350636370
1	192.168.114.72	1350636427	1350636662
1	192.168.114.73	1350636012	1350636272

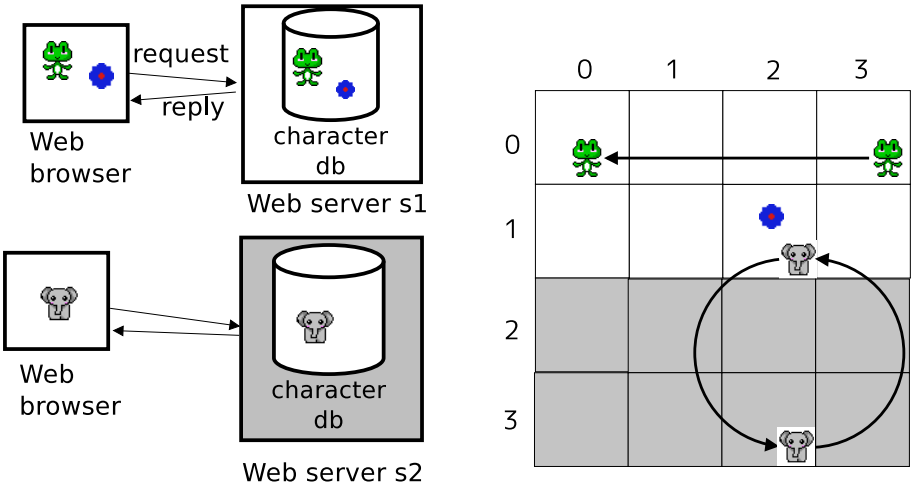
Table 4: Database Table for Time-based Rule

4.2 Time-based Rule

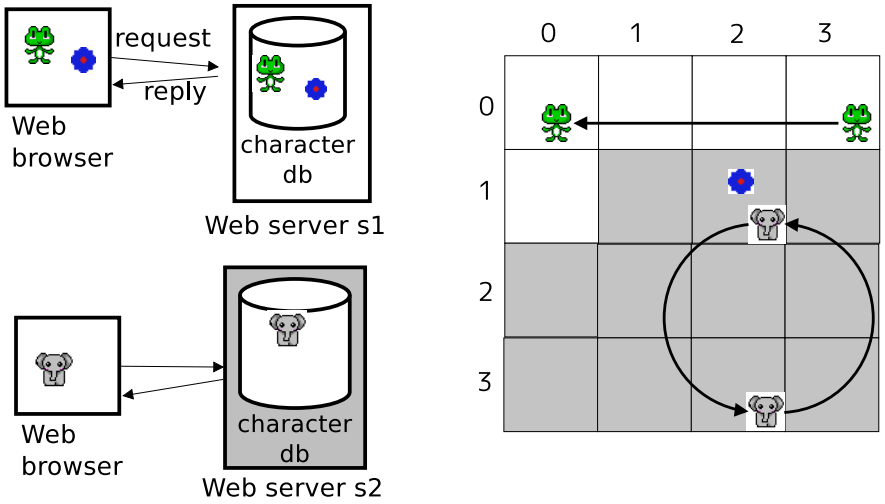
Time-based rule takes into account the amount of time an avatar spends in a block. If an avatar stays in the same block for a long time, the home for the block moves to the web server that requires the block information for the longest period. This rule is based on the concept that an avatar will continue to stay on the same block if it has already been there for a long time.

Table 4 shows an example of the database table for the time-based rule. When the CGI receives a request, it updates the start or end field of the host that is connected the web browser who requires the block. If the CGI receives a request at the first time, it updates the start field with current UNIX time. After that, the CGI updates the end field for each request. The CGI moves the home of the block to the web server which has the largest difference between the start and the end time. If the CGI does not receive a request from the web browser, it resets the start and end field for the host that connected by the browser to the value 0.

Figure 12 shows one such case. Once again, there are two web servers, s1 and s2 and two web browsers. Server s1 is home of white blocks and server s2 is home of gray blocks. The web browser for the frog connects to server s1 and that for the elephant



a) before move



b) after move

Figure 12: Time-based rule

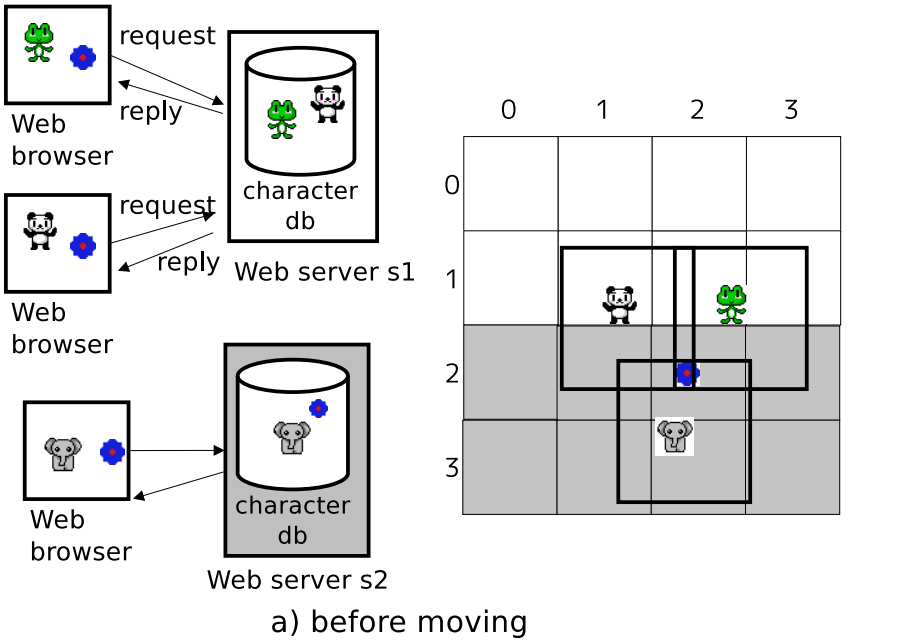
connects to server s2. The NPC is in the block (2, 1), currently home on s1. The frog moves back and forth along a straight line. The elephant moves along a circular path. This case focuses on blocks (1, 1), (2, 1) and (3, 1). Server s1 requires its blocks for a short period, because the frog passes through this area quickly. In contrast, server s2 requires its blocks for a long period, because the elephant moves in a circular path around this area. In this case, the system moves the home for these blocks from server s2 to server s1.

4.3 Count-based Rule

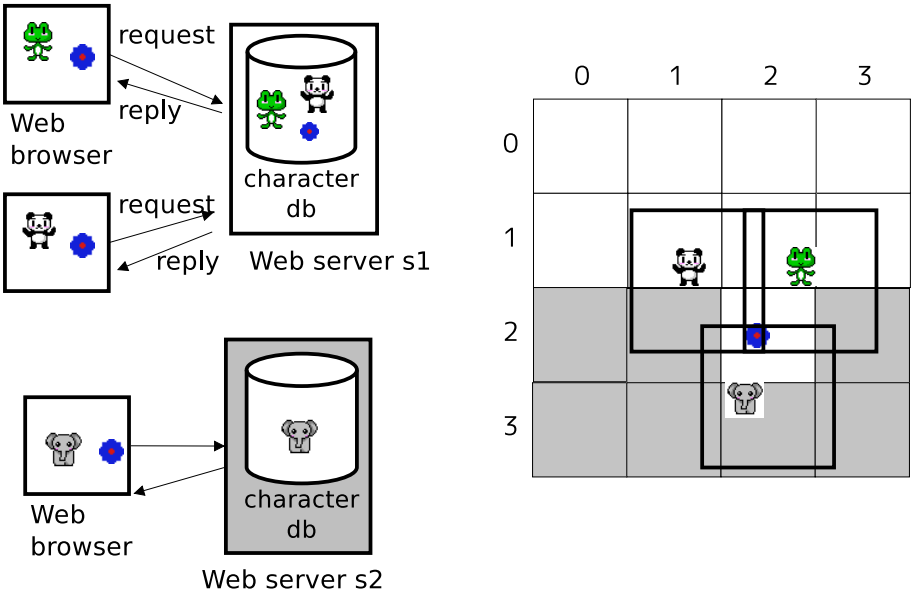
Count-based rule considers the number of avatars that requires a block. Using this rule, the home of a block is moved to the web server that serves the largest number of avatars that require the block.

Table 5 shows an example of the database table for the count-based rule. This database stores the number of avatars in a block for every host. If the avatar that connected to the host 192.168.114.70 requires the block id 0, the count field of the block id 0 for the host 192.168.114.70 is incremented. When the avatar leaves the block, the count field is decremented. The CGI moves the home of a block to a web server which has the largest number of the avatars who require the block.

Figure 13 shows an example. There are two web servers, s1 and s2 and three web browsers. Server s1 is home of white blocks and server s2 is home of gray blocks. The web browser for both the frog and the panda are connected to server s1 and that for the elephant is connected to server s2. A NPC exists in the gray block (2, 2). This block is included in the view area of all three web browsers. Then, server s1 makes a remote access to obtain the NPC information on behalf of the frog and the panda. Because there are two avatars that require the information from the gray block (2, 2) on server s1, but only one avatar that requires this information on the server s2, the home for block (2, 2) moves to server s1. After moving the home, server s1 does not



a) before moving



b) after moving

Figure 13: Count-based rule

id	host	count
0	192.168.114.70	2
0	192.168.114.71	7
0	192.168.114.72	2
0	192.168.114.73	3
1	192.168.114.70	4
1	192.168.114.71	1
1	192.168.114.72	6
1	192.168.114.73	5

Table 5: Database Table for Count-based Rule

require any remote access to obtain the NPC information, although server s2 does. As a result, the number of remote access is reduced from 2 to 1.

4.4 Performance Evaluation

An experimental evaluation is conducted to investigate average latency and player capacity.

4.4.1 Experimental Setting

Table 6 shows the runtime environment. In this experiment, there are four web servers. The entire game world is 1500×1500 pixels and divided into 15×15 blocks. The update interval is 1500 ms. This means that a web browser invokes the getinst.cgi every 1500 ms. There are 100 NPCs. The latency measurements includes the transmission delay, database access delay and access congestion delay. As we described in the section 2, a MORPG requires a latency of less than 500 ms [17]. Therefore, this experiment adjusts the threshold of the latency to less than 500 ms for each rule.

CPU	Intel(R) Xeon(TM) 2.80GHz * 2
Memory	1GB
kernel	Linux kernel-2.4.31
compiler	gcc version 3.3.2
WebServer	Apache/1.3.33
NIC	HP NC105i PCI Express Gigabit
connection	1Gbit Ethernet

Table 6: Runtime environment

4.4.2 Client Software

A client software is implemented for this evaluation. This client is a virtual client for this MORPG. It enable to implement the several hundred of web browser request.

This client reads the avatar information and web server information from the file at login time. Then, it invokes the login.cgi to retrieve the location, the block number and so on. If this client gets the location of the avatar, it creates the request message according to the avatar information. And it invokes the getinst.cgi at a regular time interval. Then, it moves the location of the avatar.

Furthermore, this client moves the avatar with two movement patterns, random walk pattern and dense crowd pattern. With the random walk pattern, avatars take a random walk by choosing a random direction and advancing 10 pixels in that direction, 10 times each. They repeat this pattern for a while, occasionally also jumping a random point. With the dense crowd pattern, 'hotspot' is defined. It is a place where many avatars gather. In actual games, avatars tend to gather around a town, a battle point or some event place. Hence, the dense crowd pattern implements each of these as a hotspot. The entire game world is divided into nine regions and each of which has a hotspot. Each avatar randomly selects one of hotspots and moves towards it. After spending some time at the hotspot, the avatar selects a new one and moves there.

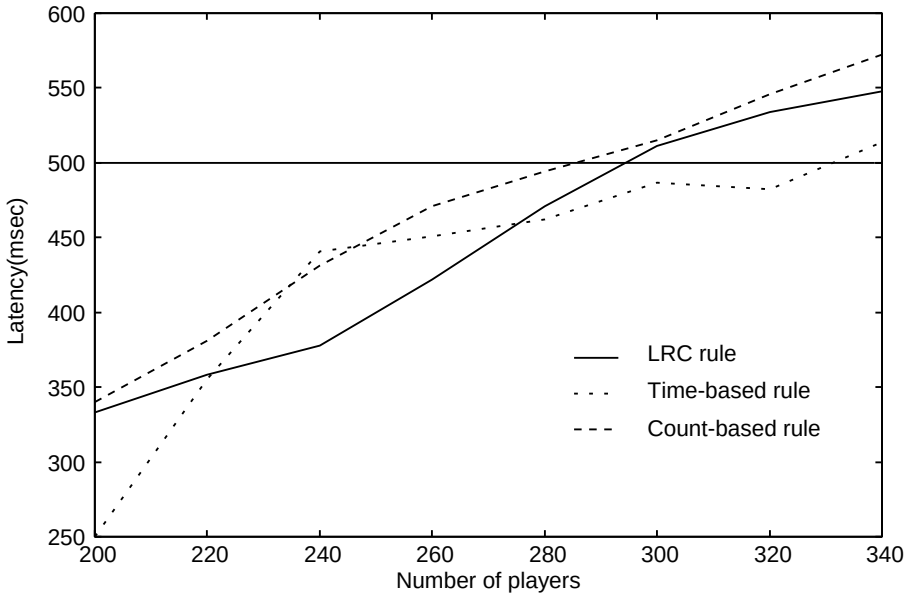


Figure 14: Latency of getinst.cgi with block allocation based on local data with random walk pattern

4.4.3 Experimental Result with Random Walk Pattern

Figure 14 shows the latency of getinst.cgi for each rule with the random walk pattern. The latency with count-based rule exceeds 500 ms when the number of avatar is 300. It is the slowest of the three. The latency with LRC rule exceeds 500 ms when the number of avatars reaches 320. Time-based rule has the best latency. It has 320 player capacity, while the single-server system has 200 player capacity. Each of LRC and count-based rule has also larger player capacity than the single-server system.

Figure 15 shows the frequency of moving home operation. The LRC rule causes the most moving home operation, whereas the time-based rule causes the least. Furthermore, the time-based rule has the stable frequency of moving home, while the LRC rule and the count-based rule has higher varied frequency of moving home.

Figure 16 shows the frequency of remote access. The LRC rule has more remote accesses than the time-based rule, even though the LRC rule causes more moving home operations. The count-based rule has the most frequent remote access. This is

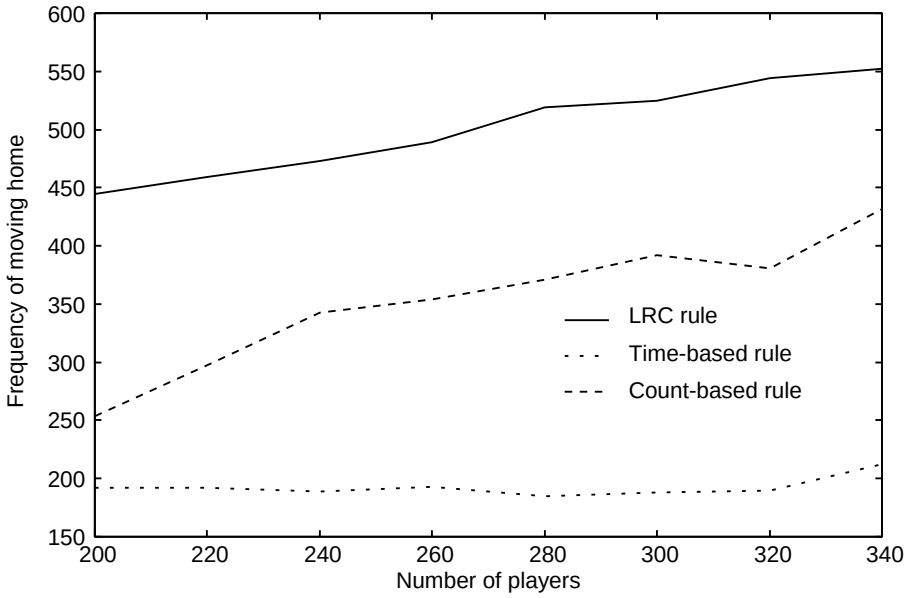


Figure 15: Frequency of moving home with random walk pattern

because that the home of the block already moved to the other web server when the getorg.cgi receives a remote access request. When getinst.cgi requires a remote access to obtain the information from a block and home of the block already moved to the other web server, the getinst.cgi needs to request a remote access to a new home of the block. This situation generates some overheads. Figure 17 shows the number of times this situation occurs. Here, the LRC rule showed the highest frequency. The frequency of the time-based rule is greater than that of the count-based rule, although the time-based rule caused the least number of moving home operations.

4.4.4 Discussion of Random Walk Pattern

The time-based rule showed the best latency. However, in terms of the number of remote accesses, the difference between the LRC rule and the time-based rule was small. This was due to the number of the database access. Hence, the number of database access for each rule was measured. The LRC rule made 25,118 database

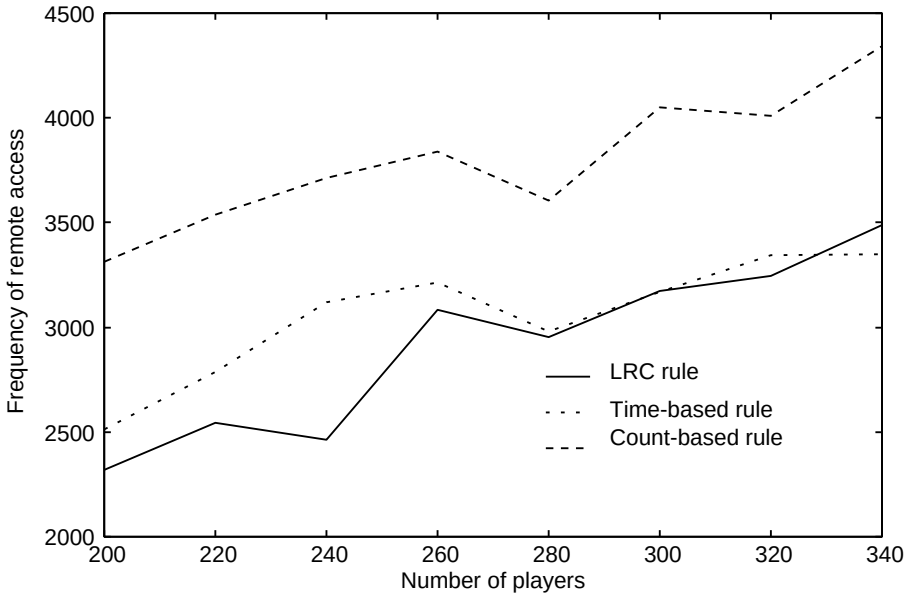


Figure 16: Frequency of remote access with random walk pattern

accesses, count-based rule made 36,307 and time-based rule made 25,118. The time-based rule accessed the database more often than the LRC rule, which resulted in more congestion in the case of time-based rule. This is because the time-based rule and the count-based rule has more complex database access pattern. Because of this database access congestion becomes a bottleneck, there was a small difference between the latencies of the LRC rule and the time-based rule.

The relationship between the frequency of moving home and that of remote access is considered. The count-based rule showed the most frequent remote accesses, although it showed more frequent moving home operation than the time-based rule. This is because the number of the web browsers connecting to each web server was balanced. In this case, moving home operations occur back and forth repeatedly. Therefore, the moving home operation does not work effectively. Then, the frequency of remote access did not change significantly, irrespective of which web server managed the block.

Finally, the relationship among the frequency of moving homes, remote accesses and the case wherein the home of a block has already moved is discussed. The case of home

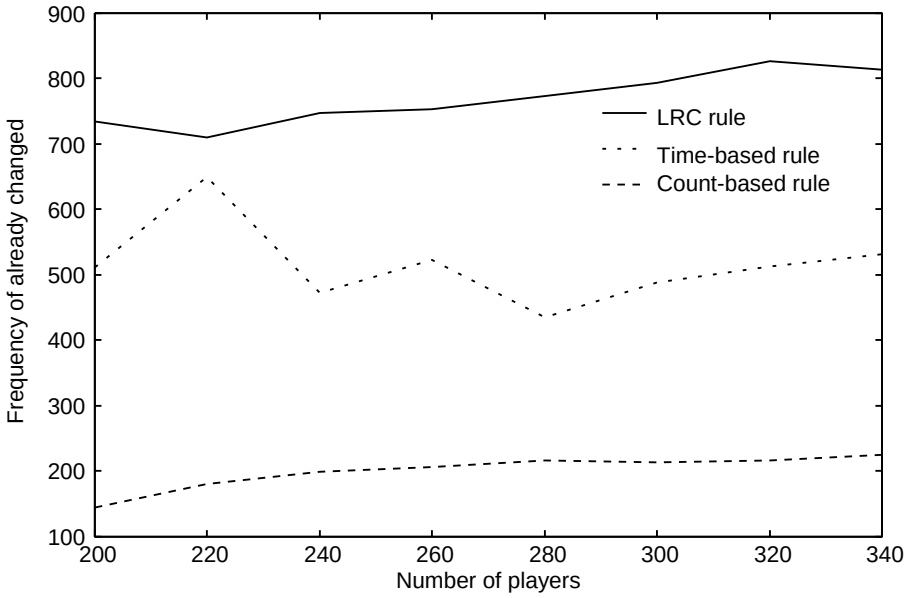


Figure 17: Frequency of already moved case with random walk pattern

blocks having already moved occurred more often according to the frequency of moving home when using LRC rule. However, for the count-based rule, the frequency did not increase as the frequency of moving homes increased. Nevertheless, the frequency of remote accesses when using the count-based rule was relatively high, even though moving home operations were occurring. The reason for this is as follows. Moving the home for a certain block may occur often if the number of requests for the block from a server is almost the same as the other servers. All avatars take a random walk, sometimes needing information from the block, and sometimes not. Each time the number of requests from s1 or s2 changes, the rule may decide to move the home of the block. This leads to frequent moving home operations, as well as frequent case in which the home has already moved.

4.4.5 Experimental Result with Dense Crowd Pattern

This experiment evaluated the three rules with the dense crowd pattern. Figure 18 shows the latency of the getinst.cgi for each rule. The count-based rule has the best latency, but the difference between each rule is small. The latency for all the rules exceeds 500 ms when the number of avatars reaches 360.

Figure 19 shows the frequency of the moving home operations. The LRC rule has the most frequent operations. The frequency of moving home operations using the count-based rule decreased as the number of avatars increased, but this did not occur when using the LRC rule.

Figure 20 shows the frequency of remote access, while the count-based rule showed the least. The count-based rule showed fewer remote accesses as the number of avatars increased, similar to the moving home operations.

Figure 21 shows the frequency of the case in which the home of a block had already moved to another other web server when the remote access occurred. This occurred most often using the time-based rule, and the least when using the count-based rule.

4.4.6 Discussion of Dense Crowd Pattern

In the case of dense crowd pattern, the difference in latency between the rules is small, although there is a difference in the frequency of remote accesses. Here, the LRC rule made fewer database accesses than the time-based rule and the count-based rule. Because database access is a relatively high load for CGI tasks, this resulted in the latencies being much the same.

The time-based rule showed the least number of moving home operations, because the dense crowd pattern let avatars gather in one block for some time. Once the gathered avatars settled in one place, there was less need for moving home operation.

Finally, the results of the dense crowd pattern are compared with that of the random walk pattern. The frequency of remote access and moving home operations using the

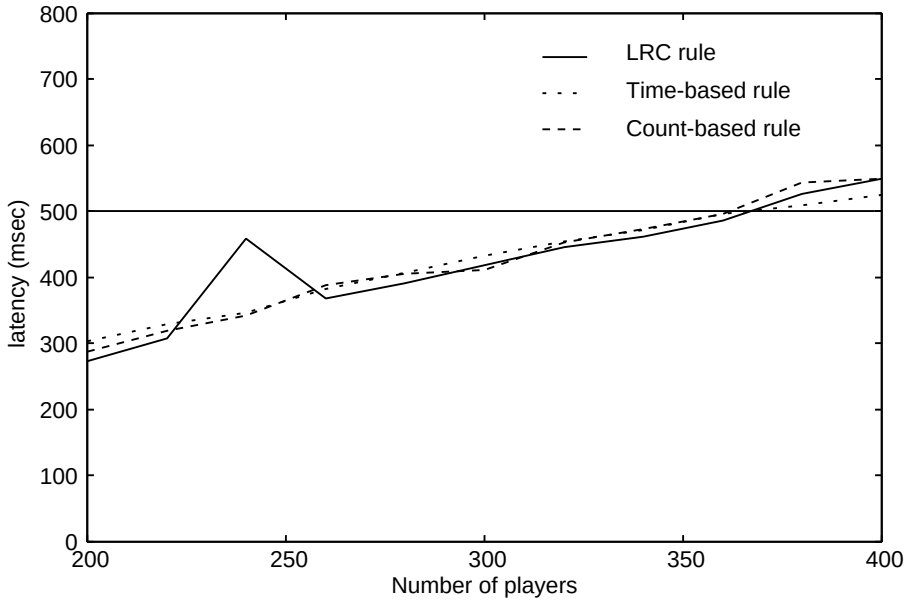


Figure 18: Latency of getinst.cgi with block allocation based on local data with dense crowd pattern

dense crowd pattern are higher than those of the random walk pattern. However, the latency when using the dense crowd pattern is shorter than that of the random walk pattern. This is due to the characteristics of the dense crowd pattern. The block required by each avatar when using the random walk pattern varies. Therefore, the CGI needs to retrieve a much larger amount of block information. In contrast, avatars within the dense crowd pattern tend to gather around a hotspot. This means that the CGI needs less blocks information. In this case, the cache mechanism works effectively, because the retrieved information is limited to that of a few blocks. As a result, the latency when using the dense crowd pattern is lower than that when using the random walk pattern.

4.5 Conclusion of Home Allocation based on local data

This section describes the home allocation based on local data. With this allocation, each web server determines which server should be a home of a block independently.

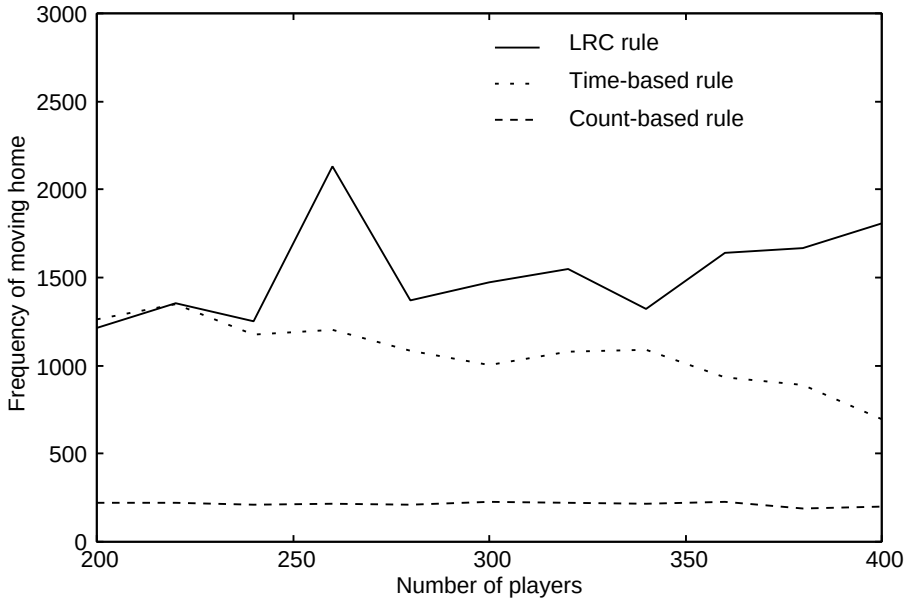


Figure 19: Frequency of moving home with dense crowd pattern

It means that servers do not have to share information related with moving home of block. To determine whether the home of the block moves or not, three rules are introduced in the system. And these rules are evaluated with two avatar movement pattern, the random walk pattern and the dense crowd pattern.

When using the random walk pattern, the time-based rule was found to have the shortest latency. It has 320 player capacity. However, it was only slightly better than that of the LRC rule. This reason is the database access conflict. The time-based rule has higher frequency of database access than the LRC rule has. Despite the small difference in latency between the time-based rule and the LRC rule, there was significant difference in the frequency of remote access between the rule. The frequency of the remote access tends to increase as the frequency of the moving home operations decreases. In addition, the frequency of situations in which the home had already moved tended to be higher as the number of moving home operations increased.

When using the dense crowd pattern, Each rule has 360 player capacity. However, there was only a slight difference in latency between the rules. In contrast, when using

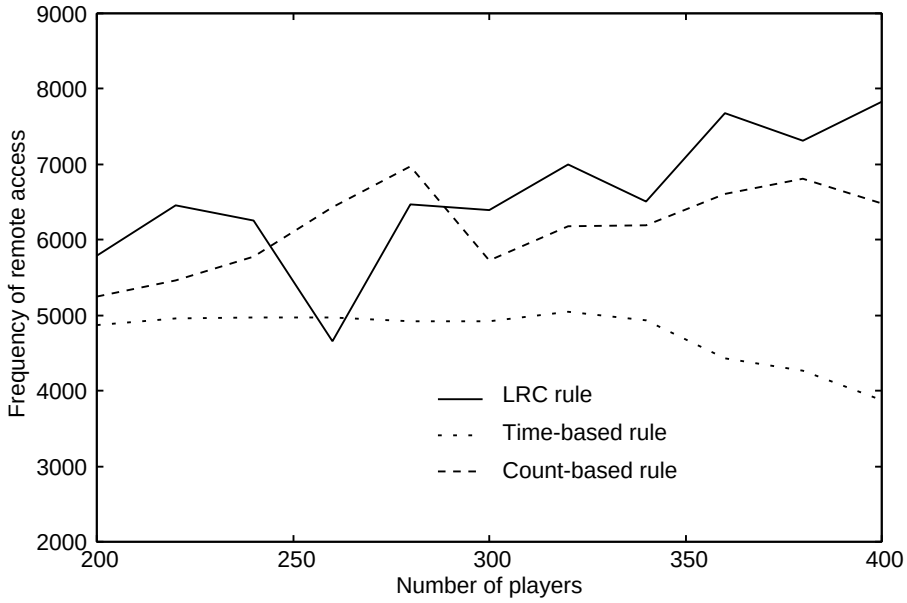


Figure 20: Frequency of remote access with dense crowd pattern

the random walk pattern, the frequency of the remote accesses tended to be higher as the frequency of moving home operation increased.

When using the dense crowd pattern, the latency is less than that of the random walk pattern. This is because the cache mechanism works effectively.

Using the home allocation based on local data, the system achieved the larger player capacity than the single-server system. However, the latency difference among servers is wide, it leads to the unfair situation. To solve this problem, the home allocation based on global data is introduced. It is described in the following section.

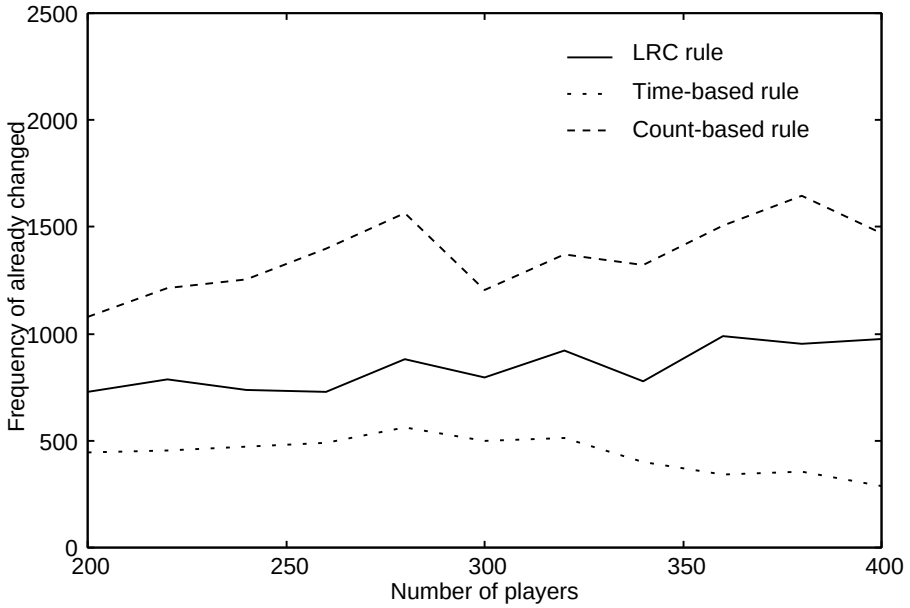


Figure 21: Frequency of already moved case with dense crowd pattern

5 Home allocation based on global data

The home allocation method based on local data could achieve large player capacity than the single-server system. However, the latency difference among servers is wide. The wider difference of server latency is, the more unfair situation is produced. This is because the game screen displayed on the web browser with the long latency is different from that with the short latency, although both of the avatars related to these web browsers are on the same region. Thus, the difference of the latency should be small. To resolve this problem, the more balancing server load is needed. The latency depends on the server load. A major factor of the server load is remote access and it is affected by the block allocation. Latency for a web browser becomes longer if some remote access is needed for the request from the browser. Thus, the home allocation that achieves the balancing the frequency of remote access is preferred.

The home allocation based on global data collects all server data. Then, this method obtains an optimal block allocation. The block allocation problem is dealt with as a combinational optimization problem. We assume that a major server load is remote access. From the view of server, there are two types of communication. Figure 22 shows these two communications. In this figure, there are three servers and two web browsers. The server A has the ownership of the block 1 and 2 while the server B has the ownership of the block 3 and 4. The frog connects to the server A and the panda connects to the server C. If the frog requires the block 3 or 4, the server A sends a request to the server B. Thus, the server A receives the information from the server B. This is one of the communication types. On the other hand, the panda sends a request to the server C. If the panda requires the block 1 or 2, the server C sends a request to the server A. Thus, the server A sends the information to the server C. This is another type of communication. For a server, the number of times of these communication is accumulated as the total server load. We aim to minimize the sum of all server loads.

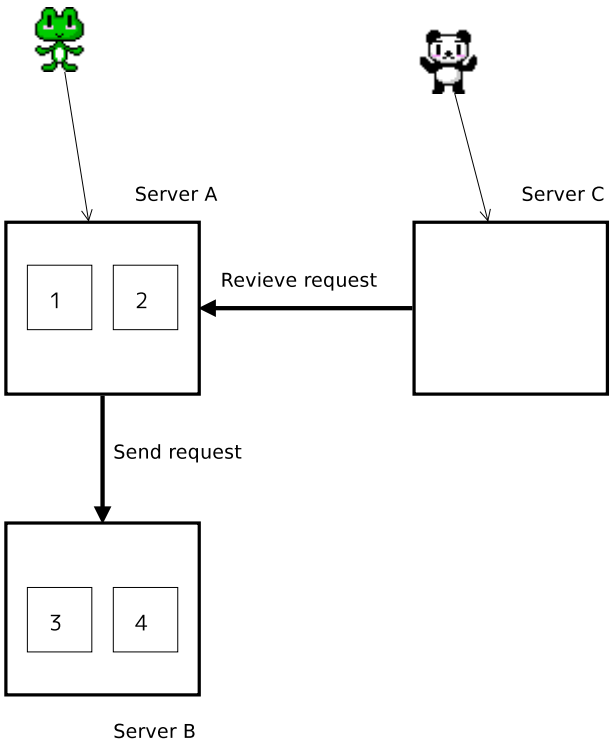


Figure 22: Two types of communication

5.1 Formulation

To solve the problem, it is formulated as a combinational optimization problem. The notations in the formulation is as follows.

S : the set of servers.

A : the set of avatars.

$A_s \subseteq A$: the set of avatars login to server $s \in S$.

B : the set of blocks.

$B_i \subseteq B$: the set of blocks included in view area of avatar $i \in A$.

x_{sb} : a value 1 if server s is the home of block b or 0 otherwise.

y_{is} : a value 1 if the view area of avatar i includes a block whose home is server s; in other words, the web browser related to avatar i needs data on server s, otherwise 0.

d_{st} : the additional load value to server s on receiving block data from server t.

u_{st} : the additional load value to server s on sending block data to server t.

w_s : an excess of load over threshold T on server s.

The x_{sb} is a decision variable. It decides which server should manage the block. If a server s manages a block b, the value of x_{sb} is 1, otherwise 0. The T is the threshold for server load as negligible latency on the fairness among players. The w_s is an excess of load over threshold T on server s. If the load is less than T, this value is zero. The objective is to minimize the sum of the value $w_s, s \in S$.

The objective function minimizes the square sum of the excess amount over T in order to maintain fairness among players. Furthermore, to highlight the difference among each server load w_s, each value of the server load is raised to power of two. This means that this problem is dealt with as a quadratic integer programming problem. The expression (1) is the objective function and the rests are constraints.

$$\text{Minimize} \quad \sum_{s \in S} w_s^2 \tag{1}$$

Subject to

$$\sum_{s \in S} x_{sb} = 1 \qquad b \in B \tag{2}$$

$$\sum_{i \in A_s} \sum_{t \in S, t \neq s} d_{st} y_{it} + \sum_{t \in S, t \neq s} \sum_{i \in A_t} u_{st} y_{is} - w_s \leq T \quad s \in S \tag{3}$$

$$y_{is} \leq \sum_{b \in B_i} x_{sb} \leq |B| y_{is} \qquad i \in A \setminus A_s, s \in S \tag{4}$$

$$x_{sb} = 0 \ or \ 1 \qquad s \in S, b \in B \tag{5}$$

$$y_{is} = 0 \ or \ 1 \qquad i \in A \setminus A_s, s \in S \tag{6}$$

$$w_s \geq 0 \qquad s \in S \tag{7}$$

(1) Minimize the square sum of the amount of server load over a threshold.

(2) Each block must be managed by exactly one web server.

(3) Threshold T indicates an acceptable server load. The sum of T and w_s is the total server load on server s. Each server should keep the lesser server load than the threshold.

(4) y_{is} must be 1 when server s is home of a block (or blocks) on the view area of avatar i. And y_{is} must be 0 when server s is not home of any blocks on view area of avatar i.

(5)..(7) are straightforward definitions.

In the expression (3), the first term indicates the total number of times that a server receives the block information from other servers. The second term indicates the total number of times that a server sends the block information to other servers.

5.2 Exact algorithm

To obtain an optimal block allocation, an exact algorithm is used to solve the above formulation. And IBM ILOG CPLEX12[33] is used to solve a problem by an exact algorithm. CPLEX12 is a multi-purpose optimization software. It solves a problem based on a branch and bound method.

IBM CPLEX reads the problem with the lp-file. Figure 23 shows an example of lp-file. It includes the objective functions, constraints and variables.

To evaluate the model, a simulation based experiment is performed. In this exper-

```
Minimize
obj: [w0^2 + w1^2 + w2^2 + w3^2 + w4^2 + w5^2]/2

Subject To
x0Bx0y0 + x1Bx0y0 + x2Bx0y0 = 1
x0Bx0y1 + x1Bx0y1 + x2Bx0y1 = 1
x0Bx0y2 + x1Bx0y2 + x2Bx0y2 = 1
x0Bx0y3 + x1Bx0y3 + x2Bx0y3 = 1
......
generals
w0
w1
w2
......
binary
x0Bx0y0
x1Bx0y0
......
```

Figure 23: Example of lp-file format

iment, the game area consists of 10×10 blocks and each block consists of 3×3 cells. These blocks are managed by five web servers. There are 50 players and each web server manages 10 players. Thus, there are 50 avatars on the game world. The avatar moves to a neighbor cell (up, down, right, left) or stays on the current cell, which is random walk pattern. This movement is determined by the uniform random number respectively at the probability of 20%. If the selected direction is out of the game area, the avatar stays. The size of view area is 3×3 blocks, which means that the view area includes 9 blocks (81 cells) in maximum. The threshold of server load T is set to 50,

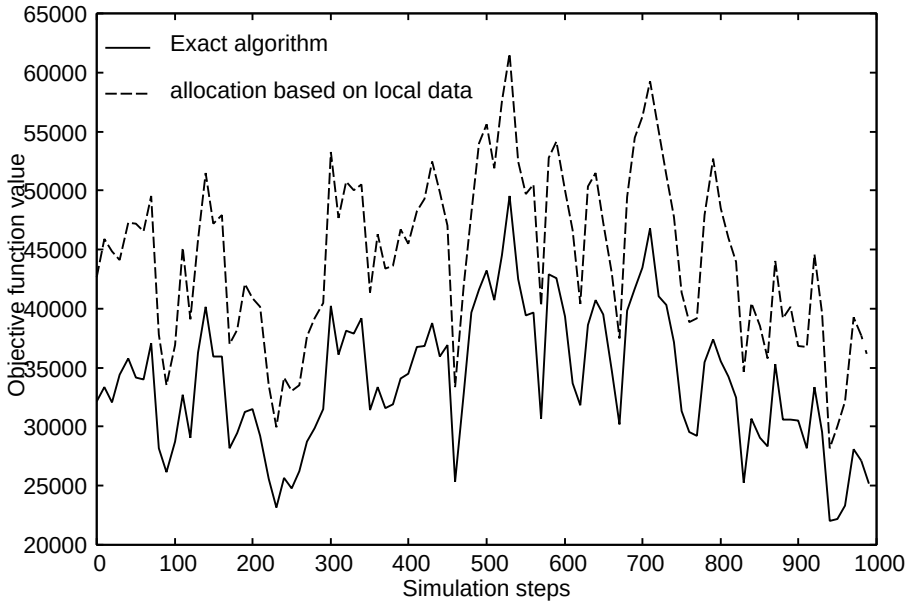


Figure 24: Comparison of stepwise objective function value of the exact algorithm and the data allocation based on local data.

d_{st} and u_{st} are set to 1.

Each problem has 705 variables (x:500, y:200, w:5) and 505 constraints. The simulation step goes forwards 1000 steps. Thus, there are 1000 problem instances.

Figure 24 shows the comparison of stepwise objective function values solved by the exact algorithm and the allocation method based on local data. The x-axis indicates simulation steps and the y-axis indicates the objective function value. All the values which are obtained by the exact algorithm are less than that by the allocation based load data. The optimal solution which is obtained by the exact algorithm has the 31% lesser average objective function value than the allocation based on local data.

Figure 25 shows the difference between the maximum and minimum objective function value in five web servers. The x-axis indicates simulation steps and the y-axis indicates the difference between the maximum and the minimum objective function value. This all values for the exact algorithm are smaller than that based on local data. The average difference is 74% lesser than that with the home allocation based

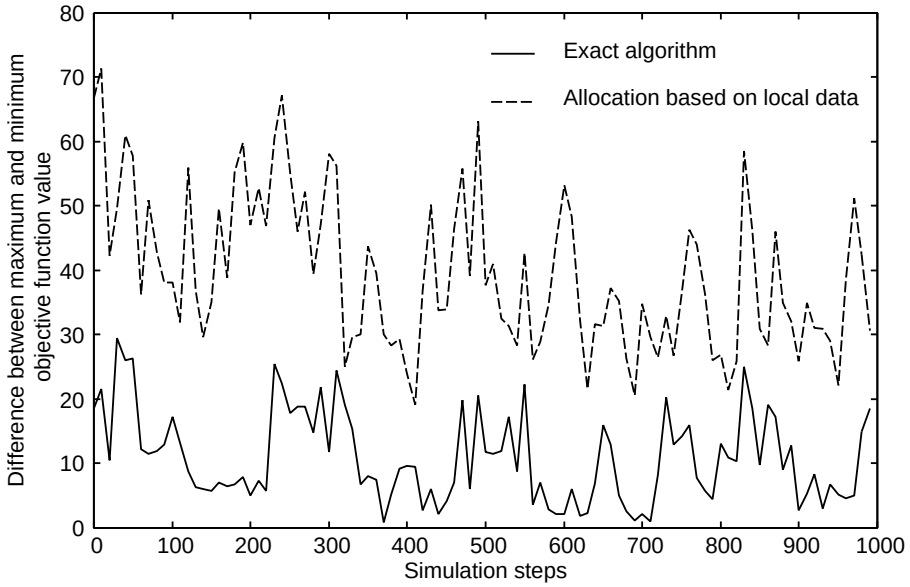


Figure 25: Difference between maximum and minimum objective function value by the exact algorithm and the allocation based local data

on local data.

Additionally, the experiment with the dense crowd pattern is performed. Figure 26 and 27 shows the result. This experiment also shows the result of value with the exact algorithm is less than that based on local data.

The exact algorithm could achieve higher fairness on both the random walk pattern and the dense crowd pattern. However, the exact algorithm took about 162 msec to calculate a block allocation for a step. Furthermore, the calculation time varies even for the same size problem because it depends on the branch and bound method. The player tolerant is 500 msec for MORPG[17]. When the exact algorithm is built into a real web-based MORPG system, the collecting data from all servers, the solving a problem, the delivering a result to servers are needed. It is preferable that the calculation time is short and uniform.

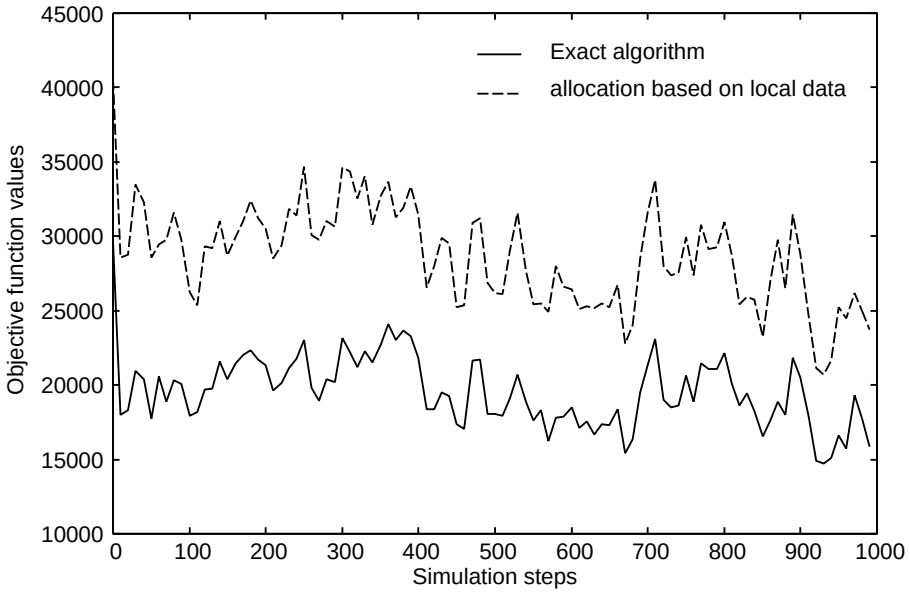


Figure 26: Comparison of server load by the exact algorithm and allocation based local data with dense move pattern

5.3 Heuristic solution approach

The exact algorithm could obtain an optimal solution. It achieved higher fairness. However, the exact algorithm took too much calculation time to solve a problem and it depends on the number of players. Thus, a technique that can obtain an optimal solution quickly is needed. Therefore, this study introduces a meta-heuristic algorithm that is based on a tabu search algorithm. Tabu search algorithm is a well-known meta-heuristic algorithm.

5.3.1 Algorithm Steps

The tabu search algorithm in this research takes the following steps.

1. as a fixed allocation, allocate blocks to each server which requires them solely.

2. allocate the rest of blocks randomly in order to share them among all servers.

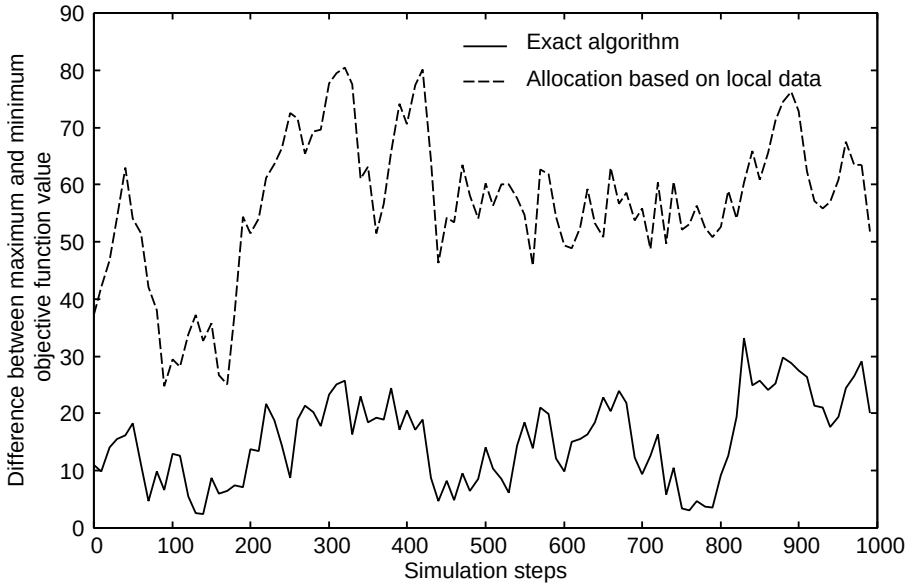


Figure 27: Difference between maximum and minimum objective function value by the exact algorithm and the allocation based load data with dense move pattern

3. hold the allocation as an initial trial solution and the incumbent solution.

4. calculate the objective function values for neighborhood solutions each of which differs in one block allocation than the trial solution.

5. select a neighborhood solution which has the lowest objective function value as the next trial solution.

6. update the incumbent solution with the trial solution if the objective function is less than the incumbent solution.

7. repeat step 4-6 for certain times.

At this time, the block which changed for a certain period does not changed.

This algorithm reads a file with the following form.

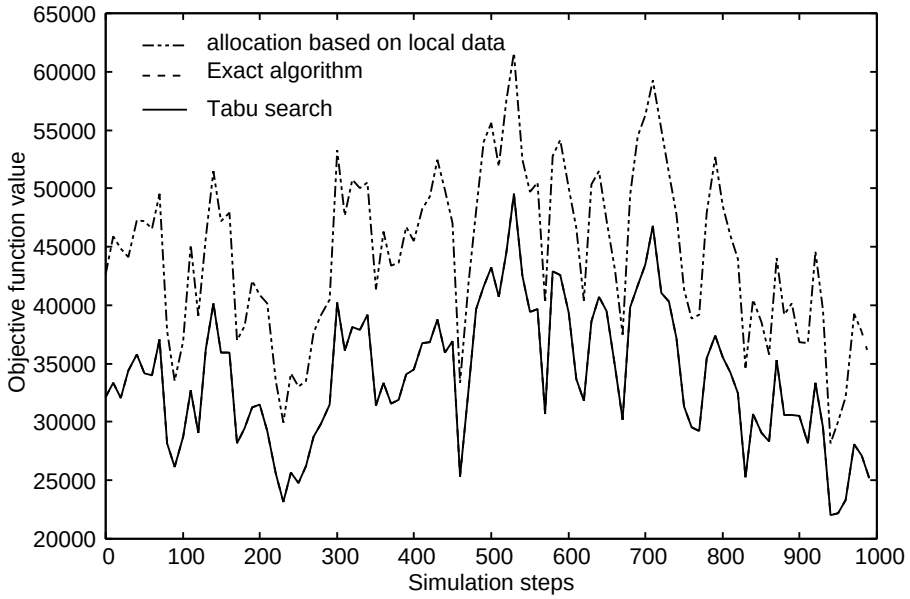


Figure 28: Comparison of stepwise objective function values of the exact algorithm, the tabu search algorithm and the data allocation based on local data

```
50


2 0 0 1 1
0 2 0 1 1
0 0 2 1 1
0 0 0 1 1
0 0 0 1 1
```

In this example, it is assumed that there are a block and five web servers. The first line shows the threshold. Each of the following line shows the load of each server in case that a server is a home of the block.

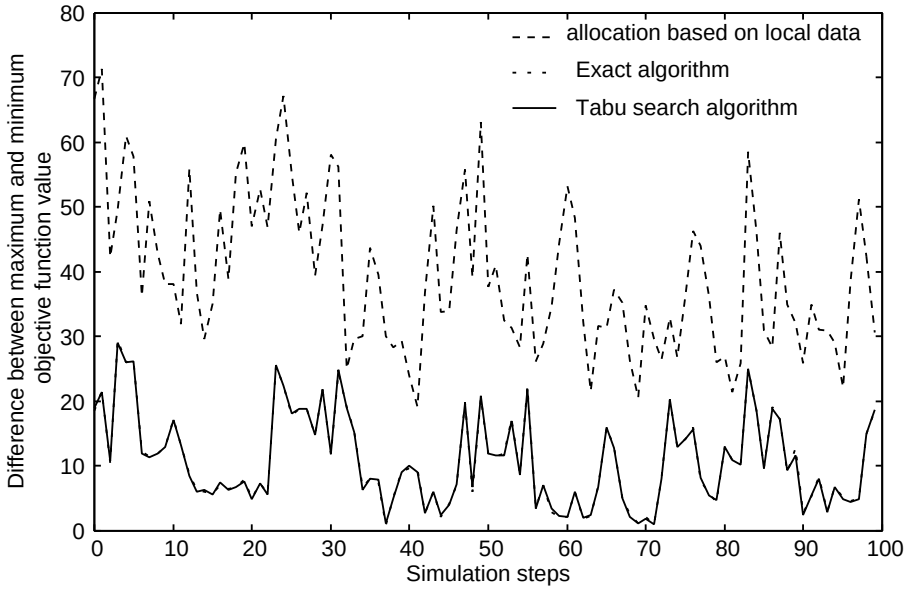


Figure 29: Difference between maximum and minimum objective function value by the exact algorithm, the tabu search algorithm and the allocation based on local data

5.4 Evaluation for Tabu Search Algorithm

To evaluate the heuristic solution approach, the simulation based experiments with the random walk pattern is performed. The experimental setting is the same as the case in the section 5.2

Figure 28 shows the comparison among the objective function values which are obtained by the exact algorithm, the tabu search algorithm and the allocation based on local data for 50 players respectively. The x-axis indicates simulation steps and the y-axis indicates the objective function values. The objective function values which are obtained by the tabu search are almost the same as the exact algorithm. It is 0.003% different from the exact algorithm. Figure 29 shows the difference between the maximum and minimum objective function values. In this figure, the values for the tabu search algorithm are also the same as the exact algorithm. Thus, the tabu search algorithm could achieve higher fairness.

As the next simulation based experiment, the number of players is changed from

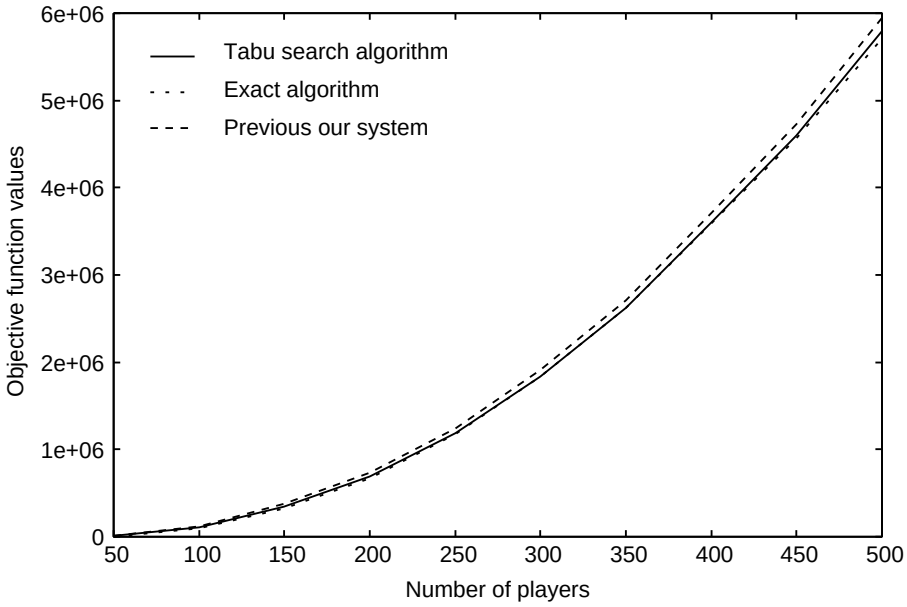


Figure 30: Comparison of average objective function values

50 to 500. Figure 30 shows the average objective function values for the number of players. For each point on x-axis, both the objective function value solved by the exact algorithm and by the tabu search are lesser than that by the allocation based on local data. The tabu search almost overlaps the exact algorithm. When 500 players, the value of the tabu search is 1% different from the exact algorithm. Thus, the tabu search algorithm could obtain a near-optimal solution.

After that, the average server load per server is evaluated. A load of server s is $w_s + T$. Figure 31 shows the average server load which serves by one web server. This shows that the result of the tabu search is almost the same as the exact algorithm. Thus, the load of each server with the tabu search algorithm is also a near-optimal solution.

In addition, fairness of the tabu search is evaluated. The wider difference leads to unfair situation. To increase fairness, the load difference among servers should be small. Figure 32 shows the difference between the maximum and the minimum server

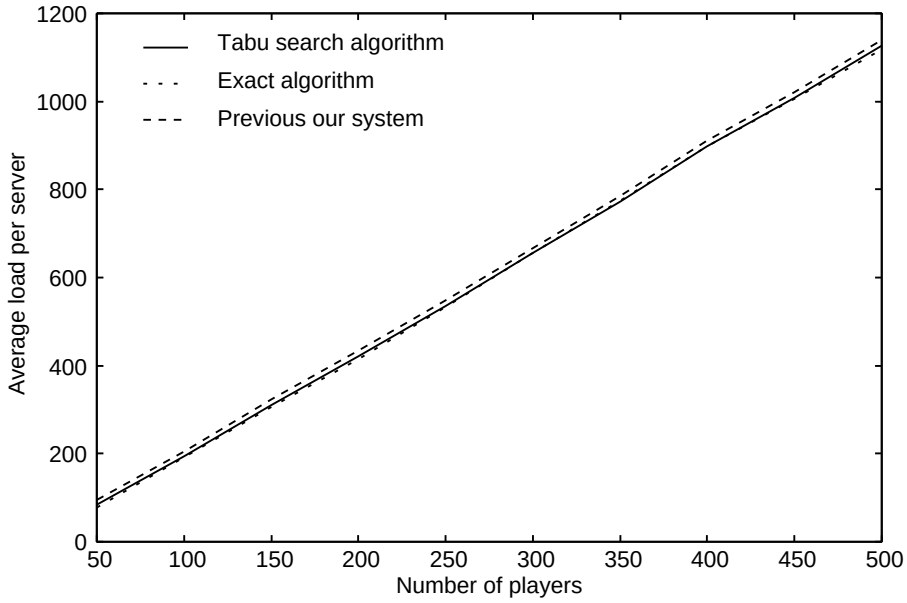


Figure 31: Comparison of average load per server

load in five servers. The result of the tabu search almost overlaps the exact algorithm. This result illustrates that the tabu search algorithm could achieve higher fairness for the number of players.

Figure 33 shows the calculation time for the exact algorithm and the tabu search algorithm. The calculation time for the exact algorithm greatly increases when 400 players. On the other hand, the calculation time of the tabu search is 25.67 msec and 29.35 msec for 50 players and 500 players. The difference between them is only slightly. This result means that the calculation time of the tabu search algorithm is stable and does not depend on a problem size. Thus, the tabu search algorithm could obtain a near-optimal solution without the increasing calculation time.

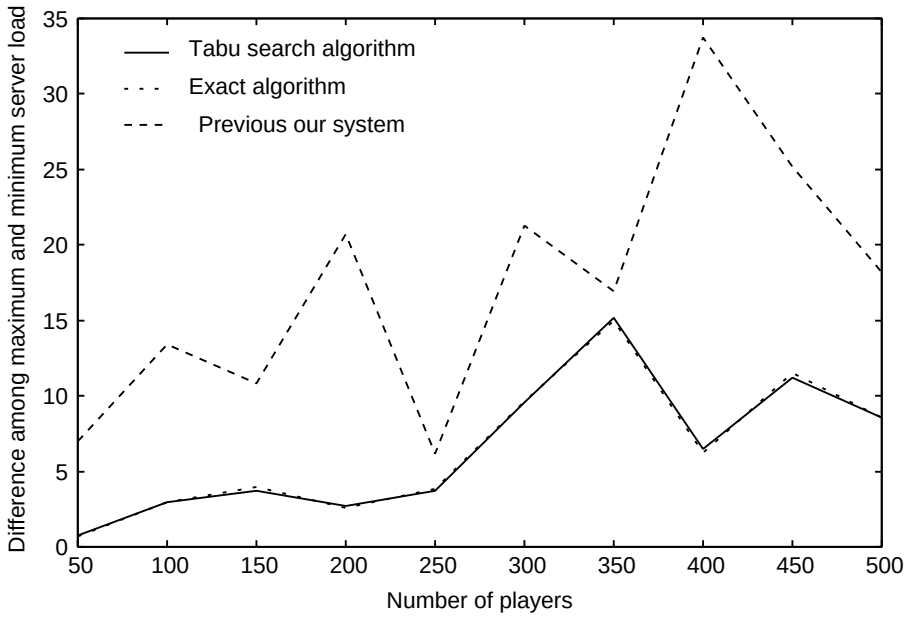


Figure 32: Difference between maximum and minimum server load

5.5 MORPG System with home allocation based on global data

The advantage of the home allocation based on global data is confirmed by the simulation based experiment. As the next step, the home allocation based on global data with the heuristic solution approach is built into our web-based MORPG system.

Figure 34 shows the system architecture for the web-based MORPG system with the home allocation based on global data. The central server collects the all data from each web server. It solves the problem with the tabu search and obtains a near-optimal block allocation. The central server solves a problem every 1500 msec. The near-optimal block allocation is delivered to all web servers and all blocks are reallocated.

Figure 35 shows the latency of servers with the home allocation based on global data. The latency exceeds 500 msec when the number of avatars is 440. This means that this system has 420 player capacity. It is the 2.1 times of the single-server system and about 1.3 times of the multi-server system with the home allocation based on

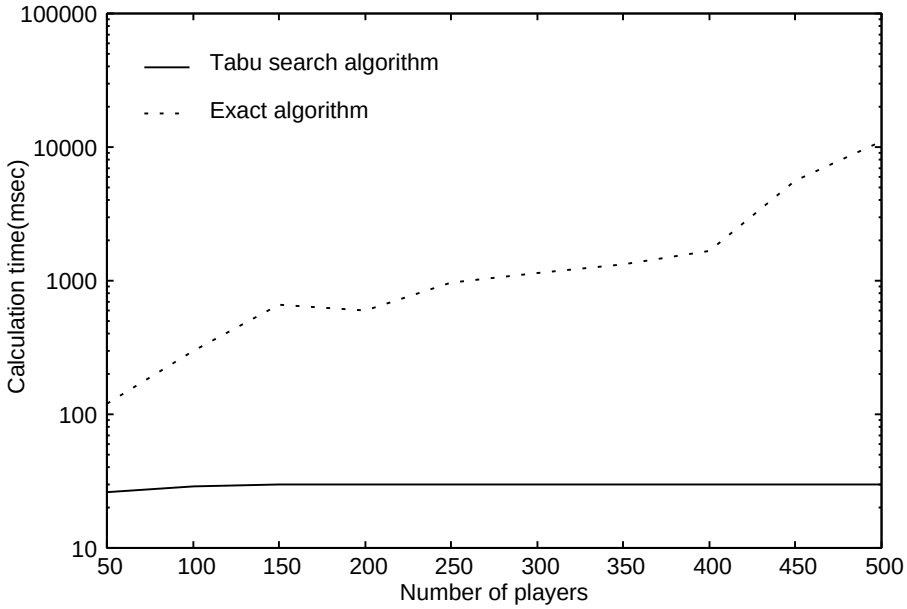


Figure 33: Calculation time for the exact algorithm and the tabu search algorithm

local data. Figure 36 shows the frequency of remote access. It is less than the home allocation based on local data. From these results, the home allocation based on global data is more effective than that based on local data.

The player capacity becomes only 1.3 times larger, although the frequency of remote access becomes 0.5 times lesser. This is because the access congestion occurs again. With 320 players, each server hosts 80 players. But each server hosts 105 players when the number of players is 420. The frequency of request to a server increases, although the frequency of remote access decreases. This issue should be solved in future works.

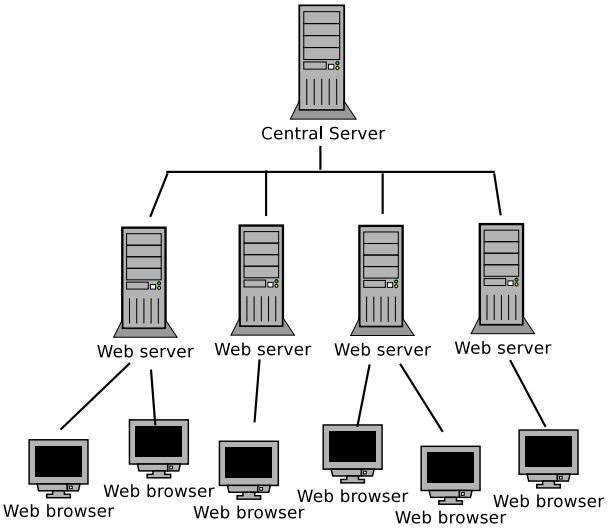


Figure 34: System Architecture for MORPG system with home allocation based on global data

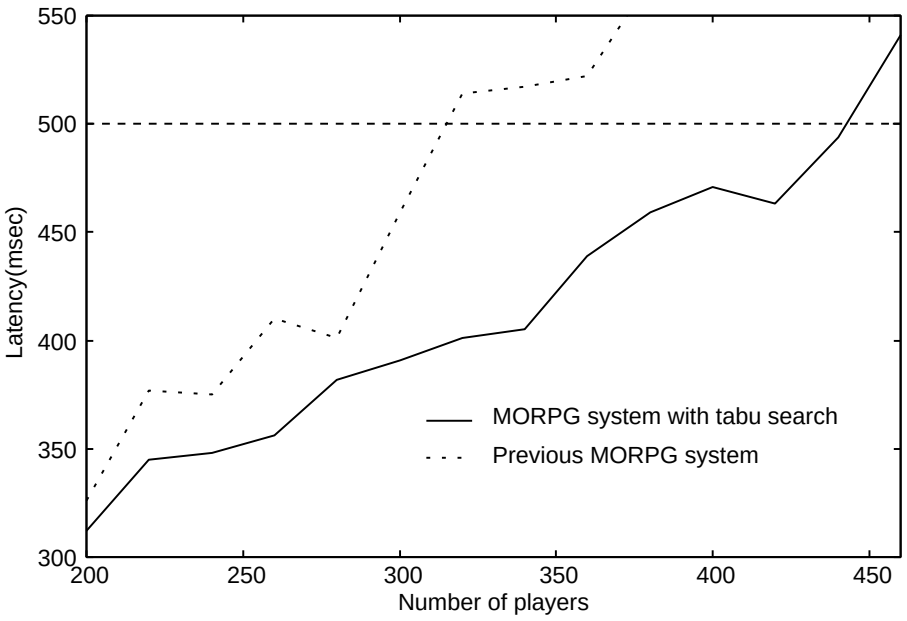


Figure 35: Latency with allocation based on global data

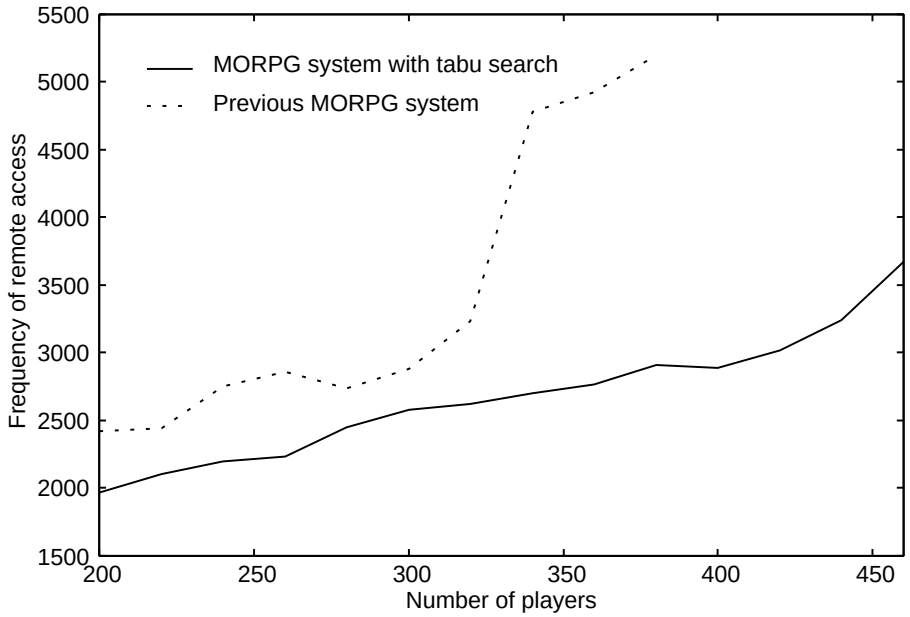


Figure 36: Frequency of remote access with allocation based on global data

6 Conclusion

This research proposed a dynamic data reallocation method for web-based online game. The access congestion to a web server is a major problem in web-based applications. It raises the database access conflict and leads the performance degradation.

To balance server load, most of other studies adopt a job migration, a resource provision and so on. In addition, they use a middleware to achieve their objective. On the other hand, this study focuses on data allocation. In this study, the web server dynamically reallocates data to balance server load. Furthermore, the dynamic data reallocation method is performed on CGI. It does not require the modifying web server and any middlewares.

The web-based MORPG system in this study divides the virtual game world into small blocks and the ownership of each block is assigned to a web server. The block data is shared among all web servers by using remote access. However, the more frequent remote access is also an overhead. Thus, a dynamic data reallocation method is introduced. Two types of methods are proposed, the home allocation based on local data and that based on global data. With the home allocation based on local data, each server independently determines which server should manage the block. To determine when the block moves, three rules are introduced, the LRC rule, the time-based rule and the count-based rule. The time-based rule achieves the most player capacity. It could serves 320 players with the random walk pattern. With the dense crowd pattern, the user capacity is 360 for three rules. However, using this method leads to the wider latency difference.

With the home allocation based on global data, all the server data is used to determine the block allocation. This allocation is dealt with as a combinational optimization problem. It is solved by an exact algorithm and a meta-heuristic algorithm. The simulation based experimental result shows that the exact algorithm could reduce the server load and achieve higher fairness. The average server load is about 31%

lesser and the load difference between maximum and minimum is about 74% lesser than the home allocation based on local data. However, the exact algorithm took too much calculation time to solve a problem. In contrast, a tabu search is possible to solve a problem quickly. It could obtain almost the same result of the exact algorithm quickly. The average server load is 0.003% difference from the exact algorithm. And the calculation time with the tabu search is stable and does not depend on a problem size. Thus, the home allocation method based on global data is built into the MORPG system. Then, it used the tabu search algorithm. It could host 420 players, it is 2.1 times of the single-server system and about 1.3 times of the home allocation based on local data.

As the future works, the model in this study should adopt a realistic situation. In the simulation based experiment, it is supposed that the overheads against both sending and receiving data are the same. However, in the realistic situation, the upload bandwidth and the download bandwidth are different. Therefore, a experiment with the realistic situation is needed. Furthermore, the speeding up for the tabu search algorithm is also needed. The decreasing calculation time makes a margin of time for other process such as the database access, the communication and the updating game data. In the MORPG system with the home allocation based on global data, a central server is needed to solve a problem. However, the home reallocation could not perform when the server crash. A MORPG system using the home allocation based on global data with out a central server is required to increase the fault tolerance.

Finally, the home allocation method might be applied to other applications. Both the allocation based on local data and that based on global data could increase the player capacity. And the allocation based on global data could achieve higher fairness. The concept of home allocation could be applied to other similar applications. One of them is a live migration of virtual machine. The live migration is a technique which migrates a virtual machine from a host machine to other host machine without shutdown it. A virtual machine is seen as a block and a host machine is seen as a home.

This is the same form as the home allocation method in this study. The applying the home allocation method to similar applications which has the same form is one of the future works.

References

[1] Daniel Pittman, Chris GauthierDickey, “A Measurement Study of Virtual Populations in Massively Multiplayer Online Games, ” NetGames’07, pp.25-30, September 19-20, 2007.

[2] Ahmed Abdelkhalek, Angelos Bilas, ”Parallelization and Performance of Interactive Multiplayer Game Servers, ” Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS’04).

[3] Jeremy Brun, Farzad Safaei, Paul Boustead, ”Server Topology Considerations in Online Games, ” The 5th workshop on Network & System Support for Games 2006-NETGAMES2006.

[4] Yi Zhang, Ling Chen, Gencai Chen, ”Globally Synchronized Dead-Reckoning with Local Lag for Continuous Distributed Multiplayer Games, ” The 5th workshop on Network & System Support for Games 2006-NETGAMES2006.

[5] Kuan-Ta Chen, Polly Huang, Chin-Laung Lei, “How sensitive are online games to network quality?,“ Communication of the ACM, Vol.49, No.11, pp.34-38, (2006. 11).

[6] Wu-chang Feng, Francis Chang, Wu-chi Feng, Jonathan Walpole, “A Traffic Characterization of Popular On-Line Games,“ IEEE/ACM TRANSACTIONS ON NETWORKING, Vol.13, No.3, pp.488-500, (2005. 6).

[7] J. Waldo, ”Scaling in Games and Virtual Worlds, ” Communications of the ACM, Vol.51, No.8, pp.38-44, (2008. 8).

[8] Radu Prodan, Vlad Nae, “Prediction-based real-time resource provisioning for massively multi-player online games,“ Future Generation Computer Systems, 25(7), July 2009, Pages 785-793

[9] Vlad Nae, Alexandru Iosup, Radu Prodan, “Dynamic Resource Provisioning in Massively Multi-player Online Games,” IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, 05 Apr. 2010.

[10] Jared Jardine, Daniel Zappala, “A Hybrid Architecture for Massively Multi-player Online Games, “ NetGames ’08, pp.60-65.

[11] Frank Glinka, Alexander Ploss, Sergei Gorlatch, Jens Muller-Iden, “High-Level Development of Multi-server Online Games, “ International Journal of Computer Games Technology, Vol.2008.

[12] Shun-Yun Hu, Shao-Chen Chang, Jehn-Ruey Jiang, “Voronoi State Management for Peer-to-Peer Massively Multi-player Online Games, “ CCNC 2008. IEEE.

[13] Bart De Vleeshauwer, Bruno Van Den Bossche, Tom Verdickt, Fillip De Turck, Bart Dhoedt, Piet Demeester, “Dynamic Microcell Assignment for Massively Multi-player Online Gamine, ” NetGames’05, (2005. 10).

[14] Bruno Van Den Bossche, Tom Verdickt, Bart De Vleeshauwer, Stein Desmet, Stijn De Mulder, Filip De Turck, Bart Dhoedt, Piet Demeester, “A Platform for Dynamic Microcell Redeployment in Massively Multi-player Online Games, ” Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video

[15] Helmut Hoyer, Andreas Jochheim, Christof Rohring, Andreas Bischoff, “A Multiuser Virtual-Reality Environment for a Tele-Operated Laboratory,” IEEE TRANSACTIONS ON EDUCATION, VOL. 47, NO. 1, pp.121-126 (2004. 2).

[16] Erik Champion, “Online Exploration of Mayan Culture,” 5th Annual International Workshop PRESENCE 2002, pp.117-128 (2002. 10).

[17] Mark Claypool, Kajal Claypool, "LATENCY AND PLAYER ACTIONS IN ONLINE GAMES, " Communications of the ACM, Vol.49, No.11, pp.40-45, (2006. 11).

[18] Fengyun Lu, Simon Parkin, Graham Morgan, "Load Balancing for Massively Multiplayer Online Games," NetGames'06 Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games.

[19] Carlos Eduardo Benevides Bezerra, Claudio Fernando Resin Geyer, "A load balancing scheme for massively multiplayer online games," Journal Multimedia Tools and Applications, Volume 45 Issue 1-3, October 2009.

[20] M. Eraslan, N.D. Georganas, J.R. Gallardo, and D. Makrakis, "A scalable network architecture for distributed virtual environment with dynamic QoS over IPv6," Proc. 8th IEEE International Symposium on Computers and Communication (ISCC '03), 2003.

[21] Herbert Jordan, Radu Prodan, Vlad Nae, Thomas Fahringer, "Dynamic Load Management for MMOGs in Distributed Environments, " CF '10 Proceedings of the 7th ACM international conference on Computing frontiers Pages 337-346, 2010.

[22] Po-Jung Huang, You-Fu Yu, Kuan-Chou Lai, Ching-Hsien Hsu, and Kuan-Ching Li, "Exploiting Dynamic Distributed Load Balance by Neighbor-Matching on P2P Grids, " 2011 IEEE Asia -Pacific Services Computing Conference, pages. 131-138, December 2011.

[23] Roman Chertov, Sonia Fahmy, "Optimistic Load Blancing in a Distributed Virtual Environment," NOSSDAV'06, 2006.

[24] Jie Zhu, Bo Gao, Zhihu Wang, Berthold Reinwald, Changjie Guo, Xiaoping Li, Wei Sum, "A Dynamic Resource Allocation Algorithm for Database-as-a-Service, " 2011 IEEE International Conference on Web Services, pp.564-571, 2011.

[25] Yi Zhao, Wenlong Huang, “Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud, ” 2009 Fifth International Joint Conference on INC, IMS and IDC, pp.170-175. 2009.

[26] Kejiang Ye, Xiaohong Jiang, Dawei Huang, Jianhai Chen, Bei Wang, “Live Migration of Multiple Virtual Machines with Resource Reservation in Cloud Computing Environments. ” 2011 IEEE 4th International Conference on Cloud Computing, pp.267-274, 2011.

[27] SecondLife, http://secondlife.com/

[28] World of Warcraft, http://www.worldofwarcraft.com/index.xml

[29] Meet-Me, http://www.meet-me.jp/

[30] RuneScape official site, http://www.runescape.com/

[31] fragoria official site, http://www.fragoria.com/

[32] ELEMENTALIA, http://www.elementalia.jp/

[33] IBM ILOG CPLEX Optimizer, www-01.ibm.com/software/integration/optimization/cplex-optimizer/

Research achievement

[34] M. Kohana, S. Okamoto, M. Kamada, T. Yonekura, “A study of load distribution technique among MORPG system, ” IEICE SIG Report, CyberWorld 11, pp.7-11, (2008.12).

[35] M. Kohana, S. Okamoto, M. Kamada, T. Yonekura, "Improving bottleneck in Web-based MORPG, " IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM'09), pp.419-424 (2009. 8).

[36] M. Kohana, S. Okamoto, M. Kamada, T. Yonekura, "A Method for Dynamic Allocation of Data on the Multiple MORPG Servers, " Forum on Information Technology 2009, vol.1, pp.423-424 (2009. 9).

[37] M. Kohana, S. Okamoto, M. Kamada, T. Yonekura, "Dynamic Data Allocation Scheme for Multi-Server Web-based MORPG System, " 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, pp.449 - 454, (2010.4).

[38] S. Okamoto, M. Kohana, "Load Distribution by Using Web Workers for a Real Time Web Application, " Proceedings of the 12th International Conference on Information Integration and Web-based Applications and Services (iiWAS2010), pp.590-595, (2010.11).

[39] M. Kohana, S. Okamoto, M. Kamada, T. Yonekura, "Evaluation of a Dynamic Data Allocation Method for Web-based multi-server MORPG System, " IEICE TRANS. INF. &SYST., Vol.E93-D,No.12, (2010.12).

[40] M. Kohana, S. Okamoto, M. Kamada, T. Yonekura, "Reallocation Criteria on Multi-Server Web-based MORPG System," 2011 IEEE 25th International Conference on Advanced Information Networking and Applications, (2011.3).

[41] S. Okamoto, M. Kohana, "Load distribution by using Web Workers for a real-time web applications, " International Journal of Web Information System, Vol.7 iss:4, pp.381-395, (2011).

[42] Y. Jimbo, M. Kohana, S. Okamoto, "A Twitter Service for School Event, "

2011 Second International Conference on Networking and Computing, pp.266-270, (2011.12)

[43] M. Kohana, S. Okamoto, M. Kamada, T. Yonekura, "Performance Impact of Virtualization on web-based MORPG System," Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services (iiWAS2011), pp.333-336, (2011.12).

[44] M. Kohana, S. Okamoto, A. Ikegami, "Optimal Data Allocation for Keeping Fairness of Online Game," 2012 IEEE 26th International Conference on Advanced Information Networking and Applications Workshops, pp.1209-1214, (2012.3).

[45] M. Kohana, S. Okamoto, T. Yonekura and M. Kamada, "Dynamic Reallocation Rules on Multi-Server Web-based MORPG System," International Journal of Grid and Utility Computing (IJGUC), 3(2/3), 136-144 (2012)

[46] M. Kaneko, S. Okamoto, M. Kohana, Y. Inayoshi, "Document Classification based on Web search Hit Counts, " Proceedings of the 14th International Conference on Information Integration and Web-based Applications and Services (iiWAS2012), (2012. 12).

[47] M. Kohana, S. Okamoto, A. Ikegami, "Optimal Data Allocation and fairness for Online game," International Journal of Grid and Utility Computing (conditional acceptance).

[48] M. Kohana, S. Okamoto, A. Ikegami, "A Meta-heuristic Approach for Dynamic Data Allocation on a Multiple Web Server System, " IEICE TRANS. INF. &SYST. (submitted).