A Meta-Heuristic Approach for Dynamic Data Allocation on a Multiple Web Server System

Masaki KOHANA\(^{(a)}\), Shusuke OKAMOTO\(^{(b)}\), Members, and Atsuko IKEGAMI\(^{(c)}\), Nonmember

SUMMARY This paper describes a near-optimal allocation method for web-based multi-player online role-playing games (MORPGs), which must be able to cope with a large number of users and high frequency of user requests. Our previous work introduced a dynamic data reallocation method. It uses multiple web servers and divides the entire game world into small blocks. Each ownership of block is allocated to a web server. Additionally, the ownership is reallocated to the other web server according to the user’s requests. Furthermore, this block allocation was formulated as a combinatorial optimization problem and solved by an exact algorithm. However, the exact algorithm takes too much time to solve a problem when the problem size is large. This paper proposes a meta-heuristic approach based on tabu search to solve a problem quickly. A simulation result shows that our tabu search algorithm can generate solutions, whose average correctness is only 1% different from that of the exact algorithm. In addition, the average calculation time for 50 users on a system with five web servers is about 25.67 msec while the exact algorithm takes about 162 msec. An evaluation for a web-based MORPG system with our tabu search shows that it could achieve 420 users capacity while 320 users greatly increases. When the number of users is 50, this exact algorithm takes about 162 msec to solve a problem. However, this exact algorithm takes too much time to solve a problem when the problem size becomes large, which includes the frequency of inter-server communication, database access, and communication among servers. The frequency of inter-server communication, database access, and communication among servers.

key words: web-based application, dynamic data allocation, tabu search algorithm

1. Introduction

The Internet technology and mobile computing environment take advanced. Then, people not only use their own personal computers but also mobile devices such as smart phones, tablets and ultra mobile personal computers to access the Internet. In addition, wireless LAN hotspots are widely spread. People are now able to access to the Internet from anywhere. Hence, users require device-independent application that can access the same information from various places such as their home, work place, and school. An example of such applications is web-based application. It becomes popular and the number of users has increased greatly. However, increasing users raises some problems. A high frequency of user requests becomes a major cause of access congestion to a web server. It leads to database access conflict and degrades the performance.

We have been conducting a study of web-based multi-player online role-playing games (MORPGs), which must serve a large number of users and a high frequency of server requests. In our previous work, we proposed a load-distribution technique for a web-based MORPG using multiple web servers [1]–[3]. This technique divides the entire game world into small blocks. And the multiple web server manages them cooperatively. However, a static block allocation leads to higher frequent communication among servers. Thus, we introduced a dynamic data reallocation technique to reduce the frequency of inter-server communication. This technique reallocates the ownership of blocks dynamically to deal with the demand of users which vary every moment. And the average latency could be decreased. User capacity with four web servers rose up to 320 users in contrast to 200 users of single server system.

As the next step, we took account of fairness among users to make the game more attractive [4]. A major factor of the latency is the frequency of the inter-server communication and it can be controlled by the block allocation. To find the optimal block allocation that achieves the highest fairness and the shortest latency, a block allocation is formulated as a combinatorial optimization problem. And the result of a simulation based experiment with an exact algorithm result shows that the optimal block allocation achieves 31% lesser server load than our previous system. However, this exact algorithm takes too much time to solve a problem when the problem size becomes large, which increases the number of users greatly increases. When the number of users is 50, this algorithm takes about 162 msec to solve a problem instance. The threshold of our MORPG is 500 msec. It includes the time for communication among servers, database access, the updating character status and calculation time. The one third of the threshold for the calculation time (162 msec) is too long. Therefore, the time to solve a problem instance should be short.

In this paper, we propose a meta-heuristic approach based on tabu search to solve a problem quickly. The tabu search algorithm could obtain a near-optimal solution quickly, although it is simple implementation. In addition, the calculation time does not depend on the problem size. A key contribution of our paper consists of two parts. The one is implementing of a real system whose engine is based on our formulation for resource allocation problem of a web-based MORPG system. The another is to confirm that the formulated problem could be solved in realistic calculation time with a tabu search algorithm on our system. The experimental result shows that our system increases the user capacity and achieves higher fairness.

The rest of this paper consists of following sections.

---

\(^{a}\)E-mail: kohana@st.seikei.ac.jp
\(^{b}\)E-mail: okam@st.seikei.ac.jp
\(^{c}\)E-mail: atsuko@st.seikei.ac.jp
DOI: 10.1587/transinf.E96.D.2645
Section 2 shows several researches related to our study. Section 3 describes the detail of our web-based MORPG system and a dynamic data reallocation technique. Section 4 gives our formulation and describes our approach based on tabu search. Finally, Sect. 5 shows our experimental results and Sect. 6 concludes this paper.

2. Literature Survey

This section introduces some researches related to our study. There are some researches for load-balancing of MORPG servers.

Fengyun et. al. proposed a technique to balance the load over the server cluster for MMORPG [5]. Their approach is done by an off the shelf NAT load balancer. It distributes the user connection over the clustered server in a round-robin fashion.

Carlos et.al. proposed a load balancing scheme for distributed MMOG servers [6]. 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.

Bart De Vleeschauwer et al provided the dynamic microcell assignment [7], [8]. They divide the virtual world to the small cells called the microcell. And they assign the microcells to a set of servers dynamically to address the higher user density which changes dynamically.

Vlad Nae et al proposed the dynamic resource provisioning method [9]. Their method provides the resource of the host to the online games based on the neural network.

Our approach is similar to the approach proposed by Bart et. al. However, we consider the inter-server communication as well as Carlos et. al. However, they takes into account the bandwidth of server node. Their algorithm is to reduce the upload bandwidth occupation of server and inter-server communication traffic. On the other hand, our approach takes into account the frequency of inter-server communication and provides fairly user experience for all users. Furthermore, the users migrate other server in the system proposed by Carlos. The users in our system connects to only one server and does not migrate, which is a characteristic of web-based application.

3. System Overview

This section describes an overview of our MORPG system. In our MORPG, a user uses a web browser as a client software. Using web browser, a user connects to a web server and creates own avatar character on the virtual game world. The user views the game screen on the web browser and operates own avatar via web browser. The avatar moves on the virtual world according to the operation. Figure 1 shows a screenshot of our MORPG. There are two kinds of characters, avatars and non-player characters (NPCs). The elephant and the frog are avatars of the users. They are operated by the own users using mouse and arrow keys. NPCs are operated by a program on a web server.

Figure 2 shows the server structure of our system. Each user connects to a web server. All web servers are mutually connected to share game information. Each web server has two cgis, getinst.cgi and getorg.cgi. Getinst.cgi handles a request from a user while getorg.cgi handles a request from other web server. Each web server has two databases, original.db and home.db. Original.db stores character information such as the character name, position, picture file and so on. Home.db stores the relation between blocks and servers. It indicates which server manages the block.

Figure 3 shows the relationship between the entire game world and the view area of web browser. The entire game world is divided into small blocks. Each ownership of block is allocated to a web server, which is called “home of block.” A home of block has responsible to update information of characters on the block and provides information to its client and other servers.

A web browser displays the game world which included in the view area. Thus, web browser retrieves the block information that included in the view area. Each web server manage some blocks. Each block is managed by only one web server. Therefore, the web server has only the original block and there is no copy of a block. Essentially, each
server runs independently. However, when the server needs a block which is managed by the other web server, the communication among servers is required. When a server receives a request from a user, it checks the blocks that are included in the view area. Then, the server retrieves the server that manages the blocks included in the view area from the home.db. If the server is home of all blocks included in the view area, the getinst.cgi retrieves the character information and replies it. On the other hand, if the view area includes some blocks that is managed by the other web server, the getinst.cgi invokes the getorg.cgi on the home of the block to obtain information. This type of inter-server communication is called “remote access.”

3.1 Remote Access

Figure 4 shows a case of remote access. In this figure, there are two web servers and two users. The user for the elephant connects to the server A while the user for the frog connects to the server B. The server A serves as home of white blocks while the server B serves as home of gray blocks. The elephant sends a request to the server A. Then, the view area of the elephant includes gray blocks. The getinst.cgi invokes the getorg.cgi on the server B to retrieve the information of the gray blocks. After that, the getinst.cgi replies. On the other hand, the frog sends a request to the server B. In this case, the view area of the frog does not include white blocks. Therefore, the getinst.cgi on the server B does not need to invoke the getorg.cgi on the server A.

3.2 Moving Home

Many remote accesses could be an overhead for the system. In addition, user demand changes dynamically. To address them, our system reallocates a home of block dynamically using a technique called “a moving home.”

A moving home changes a role of a home of a block from a web server to another according to some rules [2]. Figure 5 shows a case of a moving home. In this example, the rule defines a condition of a moving home, which depends on whether a user require a block or not. There are two web servers and a user. The server A is a home of white blocks while the server B is a home of gray blocks. The user for the elephant connects to the server A. Because the view area of the elephant includes gray blocks, the getinst.cgi invokes the getorg.cgi on the server B. However, the other users does not require these gray blocks. In this case, these gray blocks should be managed by the server A. Therefore, our system changes the home of these gray blocks to the server A. As a result, the server A does not need any remote access for these gray blocks after the moving home occurs.

4. Modeling

In our previous work, our system could increase the user capacity by using the moving home method [1]–[3]. However, the latency difference among servers is wide. The wider difference of server latency is, the more unfair situation becomes noticeable. This is because the contents of the game screen displayed on the web browser with the long latency is different from that with the short latency, although both of the users view the same region of the game world. Thus,
the difference of the latency should be small. To resolve this problem, the more balancing server load is needed. We consider that a major factor of the server load is the frequency of remote accesses. It can be controlled by the block allocation. Therefore, we would like to take an optimal block allocation that achieves the balancing server load. Because the frequency of remote access can be controlled by the block allocation, we consider the block allocation problem as a combinational optimization problem. The proposed system in this paper is almost the same as our previous system. Each web server manages some blocks and there is no copy of a block. The sharing block information is achieved by the communication among web servers that is called “remote access.” The difference points between the proposed system and the previous system are the timing of the moving home and the condition of the moving home. In our previous system, the server checks the condition of the moving home when the remote access occurs. The web server which manages the required block decides whether the block moves to the other web server or not. Then, the server uses only the local information. It means that the server decides the block allocation independently. The condition of the moving home is the number of users which require the block or how long the user requires the block. In contrast, the proposed system does not move the block when the remote access occurs. In this system, a server solve a combinational optimization problem periodically. Then, the server collects all block information from all web servers. This means that the block allocation is decided with global information. After the solving a problem, the server delivers the block allocation to all web servers and all web servers exchange the block according to it.

We assume that a major server load is remote access. From the view of server, there are two types of communication. Figure 6 shows these two communications. In this figure, there are three servers and two users. A user operates the frog and the other operates the panda. 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 user for the frog connects to the server A and the user for 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. This means that the server A sends the information to the server C. This is another communication type. For a server, the total number of times of these two types communication is the total server load. We aim to minimize the sum of all server load.

4.1 Formulation

We formulate the block allocation as a combinational optimization problem. At the first, we describe notations in our formulation.

\[ S: \text{the set of servers.} \]
\[ A: \text{the set of avatars.} \]
\[ A_i \subseteq A: \text{the set of avatars login to server } s \in S. \]
\[ B: \text{the set of blocks.} \]
\[ B_i \subseteq B: \text{the set of blocks included in view area of avatar } i \in A. \]
\[ x_{sb}: \text{a value 1 if server } s \text{ is the home of the block } b \text{ or 0 otherwise.} \]
\[ y_{is}: \text{a value 1 if the view area of avatar } i \text{ includes a block whose home is server } s; \]
\[ d_{st}: \text{the additional load value to server } s \text{ on receiving block information from server } t. \]
\[ u_{st}: \text{the additional load value to server } s \text{ on sending block information to server } t. \]
\[ w_s: \text{a excess of load over threshold } T \text{ on server } s. \]

The \( x_{sb} \) is a decision variable. It decides which server should manage a block. If a server \( s \) manages a block \( b \), the value of \( x_{sb} \) is 1, otherwise 0. The \( T \) is a threshold for server load as negligible latency on fairness among users.

The block allocation problem is formulated as follows. The expression (1) is the objective function and the rest are constraints.

Minimize \( \sum_{s \in S} w_s^2 \) \hspace{1cm} (1)

\( \sum_{s \in S} x_{sb} = 1 \hspace{1cm} b \in B \) \hspace{1cm} (2)

\( \sum_{t \in A_i} \sum_{s \in S} d_{st} y_{is} + \sum_{t \in S} \sum_{e \in A_i} u_{st} y_{is} - w_s \leq T \hspace{1cm} s \in S \) \hspace{1cm} (3)

\( y_{is} \leq \sum_{b \in B_i} x_{sb} \leq |B| y_{is} \hspace{1cm} i \in A \setminus A_s, s \in S \) \hspace{1cm} (4)

\( x_{sb} = 0 \hspace{1cm} s \in S, b \in B \hspace{1cm} \) \hspace{1cm} (5)

\( y_{is} = 0 \hspace{1cm} i \in A \setminus A_s, s \in S \hspace{1cm} \) \hspace{1cm} (6)

\( w_s \geq 0 \hspace{1cm} s \in S \hspace{1cm} \) \hspace{1cm} (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) A threshold \(T\) indicates an acceptable server load. A sum of \(T\) and \(w_s\) is a total server load on server \(s\). Each server should keep the lesser server load than a threshold.
(4) \(y_{is}\) must be 1 when server \(s\) is a 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 \(i\).

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.

4.2 Meta-Heuristic Algorithm

An exact algorithm for the above formulation could obtain an optimal solution. However, it takes too much time to calculate it depending on the number of users [4]. The user tolerant in MORPG is 500 msec or less [10]. The longer calculation time is not feasible. Thus, a technique which can obtain an optimal solution quickly is needed. Therefore, we introduce a meta-heuristic algorithm. As a preliminary investigation, we use a tabu search algorithm. This algorithm is easy to apply to our block allocation model. Using the tabu search algorithm, we could obtain a near-optimal block allocation quickly in our previous work, although the implementation is very simple. Furthermore, the calculation time does not depend on the problem size. Our tabu search algorithm 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.
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 value is less than the incumbent solution.
7. repeat step 4–6 for certain times.

5. Experimental Result

5.1 Simulation Based Experiment

At first, we conduct a simulation based experiment and compare our tabu search algorithm with an exact algorithm. And we also compare our tabu search algorithm with our previous moving home system with LRC rule. The LRC rule is a criterion to determine when the block moves from a server to the other server. The block is moves to the other server when the block is required by only one user with the LRC rule. In this experiment, the entire game area consists of 10 \(\times\) 10 blocks and a view area size is 3 \(\times\) 3 blocks. There are five web servers and 50 users. Thus, each server hosts 10 users. The number of trials is 1000, which means that our tabu search algorithm repeats the step 4–6 that is described at Sect. 4.2 for 1000 times. Each avatar chooses a direction randomly (up, down, right, left and stay) and go to the chosen direction for 10 simulation steps. The avatar moves the distance which equals one third of one side length on a block. After the 10 steps, the avatar chooses a new direction again. The threshold \(T\) is 50. The simulation step goes forwards 1000 steps. Thus, there are 1000 problem instances. Each instance has 705 variables (\(x\): 500, \(y\): 200, \(w\): 5) and 505 constraints. The \(x\) is the number of the variables \(x_{sb}\). The variable \(x_{sb}\) indicates whether the server \(s\) manages the block \(b\) or not. The \(y\) is the number of the variables \(y_{is}\). It indicates that whether the avatar requires the block whose home is server \(s\). The \(w\) is the number of the variables \(w_i\). It is the same as the number of servers. The variable \(w\) indicates a excess of server load over threshold \(T\) on server \(s\). To obtain an optimal solution solved by an exact algorithm, IBM ILOG CPLEX12 was used [4]. IBM ILOG CPLEX12 is a multi-purpose optimization software [11].

Figure 7 shows the comparison among the objective function values which are obtained by our previous moving home method, an exact algorithm and our tabu search algorithm respectively. It illustrates an objective function value of each simulation step. The \(x\)-axis indicates simulation steps and the \(y\)-axis indicates the objective function values. All the objective function values which are obtained by the exact algorithm are lesser than that by our previous method. The optimal solution has the 31% lesser average objective function value than our previous method. The solution which is solved by our tabu search algorithm also lesser than our previous method. In addition, the values of our tabu search almost overlaps with the exact algorithm. The line of the exact algorithm is hid by the line of the tabu search. The difference of the solution from the exact algorithm is only 0.003%.

At the second simulation based experiment, the num-
ber of users is changed from 50 to 500. Figure 8 shows the average objective function values for the number of users. For each point on x-axis, the objective function value solved by the exact algorithm is lesser than that by our previous method. The value with 500 users by the exact algorithm is 4% lesser than our previous method. The objective function value by our tabu search is lesser than that by our previous method on each point of x-axis. And it almost overlaps with the exact algorithm. The objective function value with 500 users by our tabu search is 1% different from that by the exact algorithm. Thus, our tabu search could obtain a near-optimal solution. In Figs. 7 and 8, we focused on the objective function values. It is the square sum of $w$, that is a excess of load over threshold $T$. As the next, we focus on the average latency per a server. A load on server $s$ is $w_s + T$. Figure 9 shows the average server load which servers by one web server. This shows that the result of our tabu search is almost the same as the exact algorithm. Thus, the load of each server with our tabu search is also a near-optimal solution.

In addition, we evaluate fairness of our tabu search. The wider load difference leads to unfair situation. To increase fairness, the load difference among servers should be small. Figure 10 shows the difference between the maximum and minimum server load in five servers. The previous method has the wider difference between the maximum and the minimum load than the other two algorithms. And the result of our tabu search almost overlaps with the exact algorithm. This graph shows that the exact algorithm and our tabu search could achieve the higher fairness.

Finally, we compare the calculation time of the exact algorithm and our tabu search. Figure 11 shows the calculation time for their algorithms. The calculation time for the exact algorithm is greatly increased when 400 users. On the other hand, the calculation time for our tabu search is stable although that is increased when 100 users. Our tabu search takes about 25.67 msec to solve an instance with 50 users in contrast the exact algorithm takes about 162 msec. With 500 users, the calculation time is about 29.35 msec for our tabu search. It is only slightly different from 50 users. This means that the calculation time of our tabu search is stable and does not depend on a problems size. Thus, our tabu search algorithm could obtain a near-optimal solution without the increasing calculation time. This is because the calculation time of our algorithm depends on the number of iterations, which is controllable by a pre-defined setting.

5.2 Real System

We confirmed the advantage of our tabu search algorithm from the simulation based experiment. Thus, we build this algorithm into our web-based MORPG system. Figure 12 shows the server structure for our tabu search version MORPG system. The central server collects all information from other web servers. Then, it solves an allocation problem at a regular intervals and obtains the optimal block configuration.
allocation. The central server delivers this allocation to all web servers. And then the servers exchange the ownership of the block according to the optimal block allocation.

In this experiment, there are four web servers and one central server. Table 1 shows the configuration of Web servers. Table 2 shows the configuration of client machine. The entire game world consists of $10 \times 10$ blocks and a view area size is $3 \times 3$ blocks. There are 100 NPCs. The central server solves our tabu search every 1500 msec. Each user uses a web browser as an own client software. Each web browser connects to a web server and sends a request to a server every 1500 msec. Each web server hosts the same number of users. It means that 50 users connect to one of four web servers when the total number of users is 200. And this experiment measures the latency for 10 minutes. Thus, each user sends totally about 400 requests. The client machines and the server machines work synchronously. And the central server and web servers work asynchronously.

We compare our MORPG system with the tabu search algorithm with our previous moving home system. Our previous system also uses four Web servers. The server configuration is the same as this experiment. It uses LRC rule. With this rule, the block moves to the other server when the block required by only one user [1]–[3]. The single server system is the same configuration as this experiment. All blocks are managed by the same web server.

Figure 13 shows the average server latency which is a time interval from the sending request to the receiving result by a client. When the number of users is 440, the latency exceeds 500 msec. Over 440 users, the latency continues to increase. Thus, the user capacity is 420 users. It is 2.1 times larger than the single-server system and 1.3 times larger than the previous our system. This user capacity is enough to the middle scale MORPG. However, it is not enough to the large scale MORPG. Therefore, the more increasing the user capacity is needed. Figure 14 shows the frequency of remote access. When the number of users is 440, the frequency of remote access is about 3300 times. It is 0.5 times lesser than the previous our system with 320 users. These results show that our system using our tabu search could reduce the frequency of remote access and shorten the average latency.

To evaluate the fairness, we calculate an unbiased variance for the latency of all requests. This value is calculated with the expression $u^2 = \frac{1}{n-1} \sum_{i=1}^{n} (\overline{x} - x_i)^2$. The $\overline{x}$ indicates the average latency and the $x_i$ is one of latency. We calculate the unbiased variance $u^2$ on the experiment for 420 users. The the value of our previous moving home system is 364577. In contrast, that of our proposed tabu search system is 231323. Furthermore, the longest latency of the proposed system is 3205 msec, although the longest latency of previous system is about 6059 msec. The tabu search system has smaller variance than the previous system. And the longest latency with the tabu search is shorter than that with our previous system. This means that our proposed tabu search system achieves higher fairness than our previous system. However, the longest latency is 3205 msec. It is not feasible because the user tolerant for MORPG is 500 msec. We need
to improve it in our future work.

The user capacity becomes only 1.3 times larger, although the frequency of remote access becomes 0.5 times lesser. We consider that this is because the access congestion occurs again. With 320 users, each server hosts 80 users. But each server hosts 105 users when the number of users is 420. Thus, the frequency of requests to a server increases, although the frequency of remote access decreases. We aim to solve this problem in our future work.

Finally, we discuss the scalability of our system. Our system could run with a large number of web servers. However, if the number of web servers increases, the frequency of remote access also increases. It raises the communication overhead. Furthermore, the calculation server must collect information from all web servers. It also raises the communication overhead. These limit the scalability of our system. To improve this problem is one of our future works.

6. Conclusion

We deal with the block allocation to web servers as a combinatorial optimization problem. We could obtain an optimal allocation by using an exact algorithm. However, it took too much time to solve a problem. In this paper, we proposed a meta-heuristic algorithm based on tabu search to obtain a near-optimal allocation quickly. Our tabu search algorithm could take a good allocation. It is about 1% different from an optimal solution. Furthermore, the calculation time of our tabu search to solve a problem is 25.67 msec while that of an exact algorithm is about 162 msec. Additionally, we build our tabu search into our MORPG system. We could achieve 420 user capacity by using our tabu search. It is 1.3 times larger than our previous system. However, the access congestion occurs again because the number of users that are hosted by a web server increases. In our future work, we need to solve this problem.

Acknowledgment

We wish to thank to the MEXT. Grant-in-Aid for Building Strategic Research Infrastructures, the Domestic & Overseas Research Program of Seikei University, Monash University and National Graduate Institute for Policy Studies. These were supporting our work.

References

[1] M. Kohana, S. Okamoto, M. Kamada, and 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, April 2010.
[2] M. Kohana, S. Okamoto, M. Kamada, and 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, pp.3173–3180, Dec. 2010.
[3] M. Kohana, S. Okamoto, M. Kamada, and T. Yonekura, “Reallocation criteria on multi-server web-based MORPG system,” 2011 IEEE International Conference on Advanced Information Networking and Applications, pp.564–570, March 2011.
[4] M. Kohana, S. Okamoto, and 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, March 2012.
[5] F. Lu, S. Parkin, and G. Morgan, “Load balancing for massively multiplayer online games,” NetGames’06 Proc. 5th ACM SIGCOMM workshop on Network and system support for games, 2006.
[6] C.E.B. Bezerra and C.F.R. Geyer, “A load balancing scheme for massively multiplayer online games,” Journal Multimedia Tools and Applications, vol.45, Issue 1-3, Oct. 2009.
[7] B.D. Vleeshauwer, B.V.D. Bossche, T. Verdickt, F.D. Turck, B. Dhoedt, and P. Demeester, “Dynamic microcell assignment for massively multiplayer online game,” NetGames’05, Oct. 2005.
[8] B.V.D. Bossche, T. Verdickt, B.D. Vleeshauwer, S. Desmet, S.D. Mulder, F.D. Turck, B. Dhoedt, and P. Demeester, “A platform for dynamic microcell redeployment in massively multiplayer online games,” Proc. 2006 International Workshop on Network and Operating Systems Support for Digital Audio and Video, 2006.
[9] V. Nae, A. Iosup, and R. Prodan, “Dynamic resource provisioning in massively multiplayer online games,” IEEE Trans. Parallel Distrib. Syst., pp.380–395, April 2010.
[10] M. Clypool and K. Clypool, “Latency and player actions in online games,” Commun. ACM, vol.49, no.11, pp.40–45, Nov. 2006.
[11] IBM ILOG CPLEX Optimizer, www-01.ibm.com/software/integration/optimization/cplex-optimizer/

Masaki Kohana was born in 1984. He received the B.Eng., M.Eng. and Ph.D.SciTech degrees from Seikei University, in 2008, 2010 and 2013. He is currently an assistant professor of Seikei University. His current research interests is load-distribution system for web-based applications. He is a member of IEICE, IPSJ, IEEE Computer Society and ACM.

Shusuke Okamoto holds B.Eng., M.Eng. and Dr. Eng. degrees all from Seikei University, in 1989, 1991 and 1994. He was a research associate (1994–1997), and an assistant professor (1997–1999) at the University of Electro-Communications. He was an assistant professor of Ibaraki University (1999–2005). He was an associate professor (2005–2011) and is currently a full professor at Faculty of Science and Technology, Seikei University, Japan. His current research interests are Web services, programming environments and parallel processing. He is a member of IEEE, ACM, IPSJ, IEICE and JSSST.
Atsuko Ikegami is a professor in the Faculty of Science and Technology of Seikei University. She received a B.S. in mathematics from Rikkyo University and she received her Ph. D from Seikei University in 2001. Her research interests are in development of models and algorithms for solving practical optimization problems. She is a member of MOS, ORSJ, IPSJ, SSJ and JSME.