ISBN 978-981-14-1455-8
Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering
WCSE_2019_SPRING
Yangon, Myanmar, February 27-March 1, 2019, pp. 132-141
doi:10.18178/wcse.2019.03.023

A Scheduling Method for Multiple Content Searching Using Dynamic Prediction of Optimum Amount of Idle Computational Resource in Large-Scale Networks

Chikara Miura [1] and Shinji Sugawara [2 +]

[1] Nagoya Institute of Technology, Japan

[2] Chiba Institute of Technology, Japan

Abstract. This paper proposes a search scheme for multiple target content items with a priority, which is [1]stored in a lot of nodes scattered in a communication network, using idle computational resources. In our previous study, our proposed scheme successfully reduced searching costs such as searching time and communication load by defining the utilization rate of all computational resources as a parameter, and setting its appropriate value according to the searching environment. In this paper, we assume that the appropriate parameter is unknown, and we propose a scheme which dynamically predicts optimal parameter value from the costs of the initial search results, and continues to make a searching schedule with correcting the optimal parameter value. The effectiveness of the proposed scheme is evaluated by computer simulations.

Keywords: content search, dynamic prediction, idle computation resource, grid computing

1. Introduction

In the recent years, network users have become to be able to exchange their contents easily over the Internet and a lot of content items have accumulated in the network because of the rapid improvement of communication bandwidth and advancement of computational processors. There is a wide variety of contents, such as texts, images, videos, music, olfactory data, haptic data and so on, and users have obtained a convenience to retrieve such contents they want easily over the network. However, it has become rather difficult to find out the rare contents they really want from a huge amount of content items scattered over the network.

These days, although it is common for the network users to utilize a search engine [1] for contents searching in the Internet environment, it is difficult to search the content items which are not connected with appropriate keywords, or which have been just released to the network right now. Then, directly searching the content sources is one of the solutions to find out the wider variety of contents from the network. This kind of method can search the contents with or without keyword connections in real-time and with flexibility, after receiving the search order (without any preparation such as crawling and indexing).

On the other hand, direct searching on the huge number of content sources over the network needs a huge amount of computational resources, and it difficult for a single ordinary network user to use the method. Therefore, this paper proposes to use a grid computing, which can aggregate a lot of small idle computational resources to supply a large computational power. Generally, the share of idle time of personal computers owned by individual users in each day is not small, so it is possible to make a large computational resource by using those idle times of a lot of PCs without the owners' disadvantages nor extra provisions of new computational resources. A lot of projects and studies of grid computing have already carried out in the fields such as gene analysis and electromagnetic waves analysis [2]. In the distributed processing of an

[+] Corresponding author. Tel.: +81 47 478 0393; fax: +81 47 478 0393.
E-mail address: shinji.sugawara@it-chiba.ac.jp.

analysis, the calculation amount of the process can be divided into some extent and it is possible to distribute the calculation load to the computational resources appropriately according to each resource's computational capacity [3]-[6]. However, in the case of using grid computing for content searching, total amount of calculation needed for a content search is unknown because the searching is executed here and there in parallel, and all the searching schedule should be discarded just after a target content is found elsewhere in the network. On top of that, because content sources and computational resources are geographically distributed in mutually independent manner in the network in general, the network load varies according to the combination that which source is searched by which resource. Therefore, the efficient scheduling method for this kind of content searching should be important.

In the past researches by the authors, some searching methods, which are available in the case where a plural number of prioritized content searching requests are issued at the same time, were proposed [7]-[10], and in each method suppressed searching times and network loads by using the utilization rate of the total computational capacity of all the resources as a parameter, and setting the parameter at an appropriate value. However, in the real network, it is usually difficult to to predict the really appropriate value by grasping the characteristic features of the network. Thus, in this paper, we propose a scheduling method which rapidly finds a plural number of differently prioritized content items requested by the users simultaneously and also depresses the network load, even in the case that the optimal value of the parameter mentioned above is preliminarily unknown. This method includes a control function to continue making a dynamic adjustment of the parameter of the following search to the optimum value in every time of the continuous searching activities based on the results of the former searches.

2. Formulation of Problem

Initially, we show the assuming environment in this paper. In the network consisting of a lot of nodes, some nodes are assumed to play a role of content sources and some are done a role of computational resources. Each content source provides its owning content items to all the users or processes of this system. Also, each user can request a plural number of content items at the same time. Henceforth, these content items are called "target content." Because the urgency of each target content's retrieval is different from the other's, each requested content item is assumed to have its searching priority. All the computational resources distributed in the network can be used by the system for the content searching, and the system efficiently makes a schedule of searching the content sources to find out the requested target content items. The specified assumptions in this paper are as follows.

1. The capacity of each content item possessed by a content source can be known previously by the system, and generally, the capacity of a content item is different from another.
2. The existence probability of a target content item in a content source is previously given independently, and can by known previously by the system.
3. A whole capacity of data possessed by a content source can be divided and searched by a plural number of computational resources in cooperation.
4. It is impossible for the computational resources involved in a cooperative searching together to change the share of searching or re-divide the capacity of the source in the process of the searching.
5. The computational capacity of each resource can be known previously by the system.
6. When a content source is searched, a single computational resource can search for only one target content item in the whole data of the source or arbitrarily divided part of the data at the same time, and a source allows two or more than two different computational resources to access the whole data or arbitrarily divided part of the data independently at the same time.
7. A user requests searching of a plural number of target content items at the same time, and their searching priorities are different in general.
8. When a computational resource finished the searching for a target in a content source and at the time of moving to the searching of the next source, the system needs a certain length of switching time to establish a connection between the source and the resource.

It is actually difficult to define the existence probability of a target content in a content source strictly, however we assume that the probability can be roughly predictable from the knowledge such as the past searching experiences, the similarity of the contents located in the same source, and so on.

Next, we will state the costs of two kinds defined in this paper. First of all, because it is obvious that the shorter the searching elapsed time from the issue of a searching request to the finish of the search is, the higher the efficiency of the searching system becomes recognized to be, the elapsed time might be treated as a cost to be reduced. When a computational resource searches for a target content item in a content source, it takes some time for applying a match search to the data possessed by the source. The length of the time can be roughly defined as the amount of the data divided by the computational capacity. Besides, when a computational resource initializes its connection to a new content source, it takes some time for switching, as mentioned before. Therefore, the total time for a searching for a single content item is assumed to be the sum of the total time of searching data at the sources and the total switching time. However, because this paper intends to treat the searching for prioritized contents, the higher the content's priority is, the sooner the content requesting user seems to want it. Then, the total time considering the users' satisfaction for searching should be one of the costs, and is defined in this paper as the product of the total searching time and the priority of the target. Note that the smaller the priority value is, the higher the searching should be prioritized. Then, the smaller the product becomes, the higher the searching efficiency concerning the time and satisfaction should be.

Secondly, network load should be another cost. When a content item is searched, data stored in the content sources needs to be retrieved by the computational resources, and this causes large amount of load in the network. There certainly are some other data must be transmitted, such as the communication within the system for the understanding of network topology, the message exchanges to notify the targets' discoveries, and so on. However, those kinds of traffic is much smaller than the data transmission between the content sources and computational resources mentioned above, and we assume that its influence to the network load can be omitted.

Two costs explained above can be defined concisely as follows.

* Cost Concerning Time and Priority

In each content searching, the product of the elapsed time and the priority is calculated and accumulated from the beginning, i.e., issuing the request of a set of prioritized content searching, to the end, that is, the situation that every content is discovered or finished to be searched in all the sources within the system. The accumulated total of the product is defined to be the cost concerning time and priority.

* Cost Concerning Network Load

In each content searching, the product of the capacity of transmitted data from the content sources to computational resources is calculated and accumulated from the beginning to the end of searching. The accumulated total of the product is defined to be the cost concerning network load.

3. Scheduling Method and Its Improvement

In this section, we firstly illustrate the scheduling algorithm used in this paper. This algorithm is the same one proposed in [10][11], which consist of two parts, such as computational resources allocation algorithm and content searching algorithm. Then, we propose its improvement.

3.1 Scheduling Algorithm

The parameters used in the algorithm is as follows.

- Capacity of data possessed by the content source n: D_n
- Existence probability of content j in the content source n: $P_b(n,j)$
- Computational capacity of the resource k: A_k
- Summation of computational capacity of the resources allocated to the content source n: S_n
- Number of computational resources allocated to the content source n: N_n
- Searching priority of target content j: $P_o(j)$

- Switching time: T
- Upper limit of computational resource utilization rate at the same time: α

Here we show detailed computational resources allocation algorithm and content searching algorithm as follows.

< Computational Resource Allocation Algorithm >

(1) Select the computational resource k_1 with the highest computational capacity among the resources which have not been allocated to any content sources.

(2) Calculate the following value from k_1, and find out the combination of the content source n_1 and target content j_1 with the highest value. j_1 in n_1 is set to be k_1's searching objective.

Note that M is a set of content sources which are planned to be searched ($n \in M$). Initially, M is an empty set.

$$\frac{P_b(n,j) \cdot P_o(j)}{D_n \cdot H(k_1,n) \cdot (1+\sum_M P_b(n,j))}$$

(3) Add n_1 searched in (2) to M.

(4) Judge if each computational resource k should be allocated to n_1 in the descending order of A_k. Firstly, calculate the following function to find out its maximum value to all the combinations of unsearched

$$\frac{P_b(m,j) \cdot P_o(j) \cdot A_k}{D_m \cdot H(k,m) \cdot (1+\sum_M P_b(n,j))}$$

content sources m and target content j.

And then, if the calculated maximum value is lower than the calculated value of the following function, the computational resource k is allocated to n_1.

(5) Judge if each computational resources allocated to content sources should be re-allocated to n_1 in the ascending order of the remaining time for current searching. Firstly, calculate the following function

$$\frac{P_b(n_1,j_1) \cdot P_o(j_1)}{H(k,n_1) \cdot (1+\sum_M P_b(n,j_1))} / (\frac{D_{n_1}}{A_k+S_{n_1}} + T \cdot N_{n_1})$$

(the same with the upper function in (4)) to find out its maximum value to all the combinations of

$$\frac{P_b(m,j) \cdot P_o(j) \cdot A_k}{D_m \cdot H(k,m) \cdot (1+\sum_M P_b(n,j))}$$

unsearched content sources m and target content j.

And then, if the calculated maximum value is lower than the calculated value of the following function, the computational resource k is allocated to n_1.

(6) If the number of allocated computational resources reaches the upper limit α, start the searching. Otherwise, go back to (1).

$$\frac{P_b(n_1,j_1) \cdot P_o(j_1)}{H(k,n_1) \cdot (1+\sum_M P_b(n,j_1))} / (\frac{D_{n_1}}{A_k+S_{n_1}} + T \cdot N_{n_1} + T_x)$$

< Algorithm Ends >

< Content Searching Algorithm >

(1) Search the target contents according to the searching schedule.

(2) When a target content is found, terminate searching and cancel the searching schedule of the same target content.

(3) If all the simultaneous searching for the target contents is finished, terminate this algorithm.

(4) Check the number of working computational resources. If the number reaches the upper limit, go back to (1) to continue searching. Otherwise, activate the process of *Computational Resource*

Allocation Algorithm to allocate sufficient number of computational resources. Then, come back here and go back to (1).

< Algorithm Ends >

3.2 Proposal of Parameter Value Prediction

In the scheduling method mentioned above, the upper limit of the computational resources utilization rate at the same time: α, is used for the resource allocation. In the proposed improvement, that is, the optimum α prediction, the initial searching costs are memorized under the conditions where all the assumable values of α are given, and based on the costs, the optimal value of α for the next search is predicted. Then, this process is repeated according to the users' continuous searching requests.

The detailed algorithm of the prediction is shown below. The parameters of *x*, *a*, *m*, and *K* are used and the meanings are as follows. The number of assumable steps of α is *m*, e.g., in the case where α can be set as 20%, 40%, 60%, 80%, and 100%, *m* is 5. *x* is a variable of natural number from 1 to *m*, *K* is a positive constant, and *a* is the number of searching times.

< Optimum α Prediction Algorithm >

(1) The total searching cost is calculated respectively for each step of α, and each of the costs is memorized as its initial "average cost."

(2) The average cost values corresponding to α are sorted in ascending order, and in this order, each α is given a natural number from 1 to *m*, i.e., α with the smallest searching cost is given the number 1, and that with the largest searching cost is given the number *m*. As a result, a set of combinations consisting of α and *x*, i.e., (α, *x*), for example, {(20%, 5), (40%, 4), (60%, 3), (80%, 1), (100%, 2)}. Note that *x* is one of the values from 1 to *m*.

(3) The following function *f*(α) is calculated by giving *x* which is combined with the value of α, and means the probability that α is selected as the utilization rate of computational resources in the next search.

$$f(\alpha) = \frac{x^{-\frac{a}{K}}}{\sum_{n=1}^{m} n^{-\frac{a}{K}}}$$

(4) The searching with the upper limit of utilization rate α decided in (3) is actually executed.

(5) The total searching cost is memorized as that in the case of using above α, and the average cost of the α is recalculated and updated using both the former memorized costs and the latest one. Then, go back to (2).

< Algorithm Ends >

In this prediction algorithm, the probability distribution described in step (3) shown above is controlled by the parameter *K*. The larger the parameter *K* is set to be, the flatter the shape of the Zipf distribution becomes and also the wider the variety of utilization rate candidates becomes. In this case, α is selected as a predicted optimum value from multiple options. Conversely, the smaller *K* is set to be, the steeper the shape of the distribution becomes and the utilization rate which makes the total searching cost the smallest in the former search tends to be selected again in the following searches, too.

4. Evaluation

In order to evaluate the efficiency of the searching method we improved by adding the parameter value prediction proposed in this paper (proposed method), we compare the performance of the method to that of the other conventional content searching methods shown below.

4.1 Methods to Be Compared

The conventional methods are briefly introduced as follows. The detailed algorithms of them are illustrated in [11].

4.1.1 Cluster Scheduling

This scheme consists of two algorithms, i.e., resource clustering and resource reservation. In the former algorithm, a plural number of geographically neighboring computational resources are grouped as a cluster, and a content source is assigned to the cluster for the cooperative searching by the cluster members. In the latter algorithm, the search reservations are preferentially made in order from the targets that are close in distance and have high existence probability and searching priority.

4.1.2 Random Cluster Scheduling

This scheme is just the same with the cluster scheduling except the way of resource's selection for the clustering. The available resources are selected to form a cluster at random from the entire network without considering their geographical locations.

4.1.3 TOS (Time cost Oriented Scheme)

This scheme is designed to minimize the elapsed time of searching. The product of the existence probability and the priority of each target content divided by its hosting source's whole data capacity is calculated in every content source, and the target contents are searched in descending order of their calculated value. Especially in the case where the switching time is assumed to be zero, the elapsed time becomes to be minimum.

4.1.4 COS (Communication cost Oriented Scheme)

This scheme is designed to try to minimize the network load. The existence probability of each target content divided by its hosting source's whole data capacity is calculated in every content source, and the target contents are searched in descending order of their calculated value. Each searching is executed by using only a single computational resource located the nearest to the source. If there are plural number of the resources with the same minimum distance to the source, all the resources are used for the searching.

4.2 Evaluation Method and Result

The evaluation is executed by using computer simulations. The environment of the simulations is shown in Table 1.

Table 1 Environment of Simulations

Parameters	Values
Number of Simulation Runs	1,000
Network Topology	BA Model
Total Number of Nodes	10,000
Number of Content Sources	1,000
Data Capacity of Content Source	10 [GB] – 1 [TB]
Number of Computational Resources	100
Computational Capacity of Resource	10 [MB/min] – 110 [MB/min]
Searching Priority of Target Content	1 - 10
Existing Probability of Target Content	0.01 – 0.10
Switching Time	1 [min]

The nodes which are neither content sources nor computation resources are simply relay nodes to form a large network. The capacity of the content source is assumed to be such as a large scale database or a website, the values of which are given from 10 [GB] to 1 [TB] at 10 [GB] intervals. The computational capacity of a computational resource is set as the amount of data capacity the resource executes per minute as the result of match search, given from 10 [MB] to 110 [MB] at 10 [MB] intervals. The target existence probability is given as the values from 0.01 to 0.10 at 0.01 intervals. The data capacity of the content sources, the computational capacity of the computational resources, and the target existence probability are assumed to be given according to Zipf distribution, that is, most of the values are very small and mutually similar, and only some limited ones are large. The priority of the target contents is assumed to be given according to power low, the range of which is from 1 to 10.

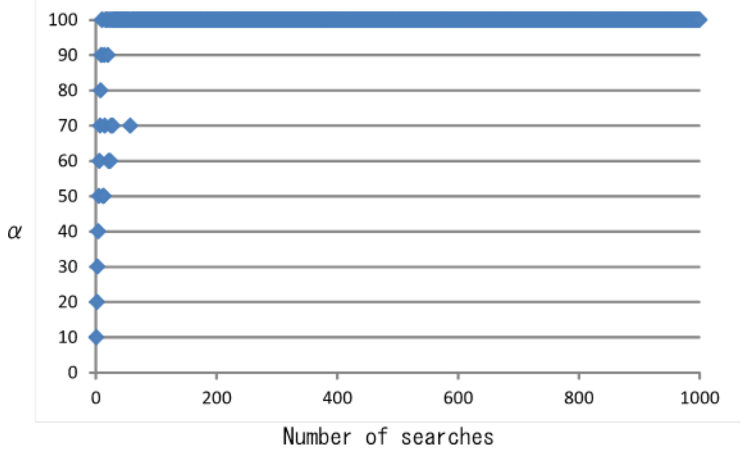

Fig. 1 Relationship between Number of Searches and Resource Utilization Rate α (K=10)

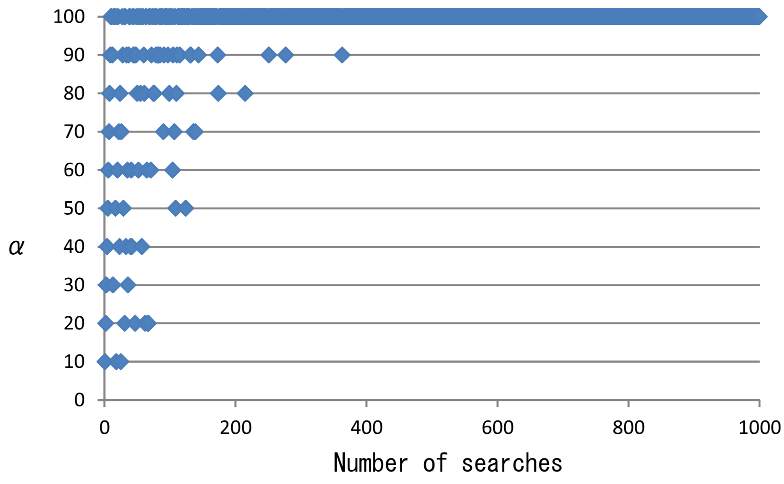

Fig. 2 Relationship between Number of Searches and Resource Utilization Rate α (K=100)

Two kinds of costs mentioned in section 2 are different each other, and they cannot simply be added together. In order to evaluate the efficiency of the searching methods, we normalize the values of the two costs based on TOS, which is the simplest method, and then, two kinds of costs are weighted respectively and added together to treat as a total cost. In that normalization, the values of both two kinds of costs, when 100 target contents are searched, are set to be the same reference value: 100 respectively, and the other values of both kinds are converted based on the reference value. The number of simulation runs is set to be 1,000, and the average value of the total costs is compared for the evaluation. In the simulation, the value of α can be set to from 10% to 100% at 10 % intervals.

When α is predicted, K is a key value to vary the average total cost. Then, we compare the dispersions of α. In the cases of K=10 and K=100, the simulation results are shown in Fig. 1 and 2. From Fig. 1, in the case that K is comparatively small, α varies little except in the early period of searching. Especially after 100 times of searching, no α value except 100% is selected as the predicted optimum utilization rate of the computational resources. On the other hand, we can see from Fig. 2 that α varies more widely in the case that K is set to be rather larger, especially in the range of the number of searching times from zero to 300. From the results, we find that the larger the parameter K is set to be, the wider the utilization rate α varies, in general.

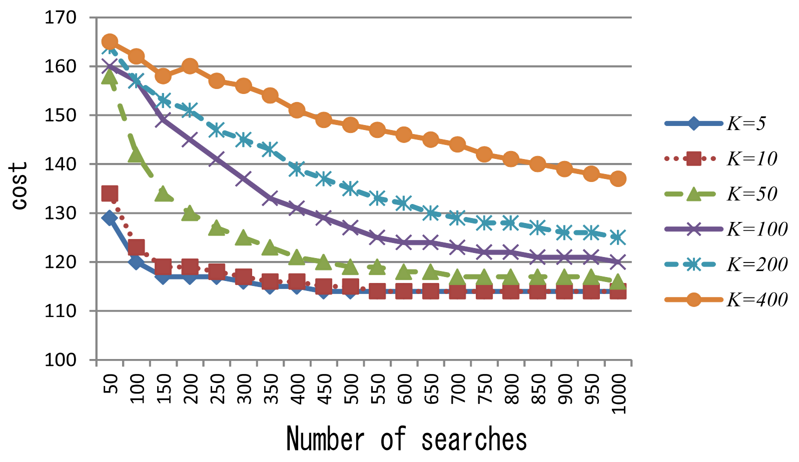

Fig. 3 Relationship between Number of Searches and Average Total Cost (when K is varied)

Next, we obtain the total searching costs of all the methods such as proposed and compared ones by computer simulations changing parameter *K*. Note that weight ratio of the two kinds of costs is 1:1. Fig. 3 shows the simulation result which illustrates the relationship between the number of searches and the total costs. We can see in the figure that the smaller the parameter *K* is, the more successfully the total cost is suppressed especially after the searches repeated around 1,000 times, because when a small value is selected as *K*, α is decided from the limited number of candidates with which rate the searching cost was successfully suppressed in the past searches, and actually the next searching cost tends to be suppressed well again. Conversely, when the parameter *K* is set to be larger, the number of α candidates becomes also larger and the predicted optimum α is narrowed down from wide variety in accordance with the progress of the searching. However, because rather large number of non-optimum searches are included until the end of the searching, the total cost becomes larger.

Generally, when *K* is set to be small, the probability to select α which successfully suppressed the total cost in the past searches, but ironically, there is a possibility to continue inappropriate α selections if an inappropriate α is selected at the earlier stage of searching, i.e., when the number of search *a* is small, because the selection tends to be maintained. When *K* is set to be too large, α narrowed down from a lot of candidates can be expected be appropriate value, but in exchange for the benefit, the total cost tends to become large. In this evaluation result, the total cost is suppressed better in the cases of setting *K* to be 5 or 10, to the same extent. Because the searching with large *K* can predict the more appropriate resource utilization rate, 10 is the most reasonable value for *K* in this case.

Setting *K* to be 10 as the best value, we then, compare the effectiveness of the proposed method to that of the methods to be compared with changing the weight ratio of the two kinds of costs. The results are shown in Fig. 4 through 6. The weight ratios between cost concerning time and priority and cost concerning network load are 1:1, 10:1, and 1:10, respectively.

Fig. 4 shows the result in case that the weight ratio is set to be 1:1. Because the narrowing down of the optimum α from the candidates is done in the early stage of the searching until around 50, the average total cost is slightly large. After the predicting value of optimum α is almost settled down, the cost of every method is reduced and the proposed method successfully suppressed the cost the best among all the methods in the period from 50 to 1,000 times of search, which is about 10% smaller than the second best method of cluster scheduling.

In the case that the weight ratio is set to be 10:1 shown in Fig. 5, the proposed method suppresses the average total cost the best to almost the same extent with the cluster scheduling. Because the cluster scheduling executes searching on the cluster basis, and all of the computational resources are always utilized, which causes the reduction of the searching time. In the situation where the cost concerning time and priority is much heavily weighted like this case, the difference of total costs between this and proposed method decreased compared with the case that weight ratio is set to be 1:1.

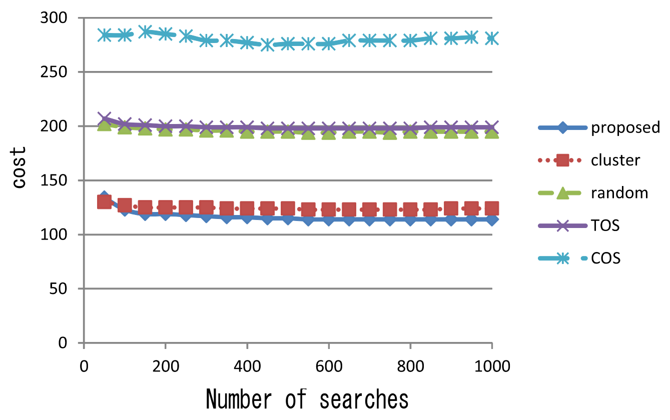

Fig. 4 Relationship between Number of Searches and Average Total Cost (weight ratio = 1:1)

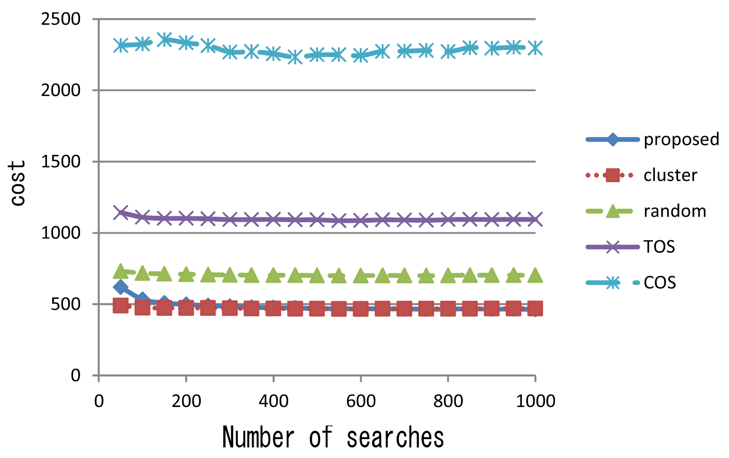

Fig. 5 Relationship between Number of Searches and Average Total Cost (weight ratio = 10:1)

In the case that the weight ratio is 1:10, the result of the evaluation is shown in Fig. 6. In this condition where the cost concerning network load is much heavily weighted, the proposed method also successfully suppressed the total cost the best to almost the same extent with COS. COS can largely reduces the network load because it always utilize only limited number of computational resources located very close to the content sources to be searched. On the other hand, the proposed method selects suppressed the computational resources utilization rate and tries to omit the useless operations in order to suppress the average total cost.

All the results of the evaluations stated above, we can say that the proposed method improved in this paper can be sufficiently effective by predicting and dynamically applying the appropriate resource utilization rates, even in the cases of changing the weight ratio between the two kinds of costs.

5. Conclusion

In this paper, we improved the scheduling algorithm in the conventional content searching method with idle computational resources distributed in large-scale network. In the improvement, the computational resources are allocated to the searching appropriately, and the upper limit of the computational resources' utilization rate is dynamically controlled by the prediction of optimal rate, in order to minimize the searching cost caused mainly from the elapsed time and the network load.

The effectiveness of the improvement was evaluated by computer simulations, which showed that the proposed method successfully reduced the cost sufficiently in the environment we set this time. As a future work, we need to evaluate the proposed method further in much more realistic environment, in order to modify the proposal.

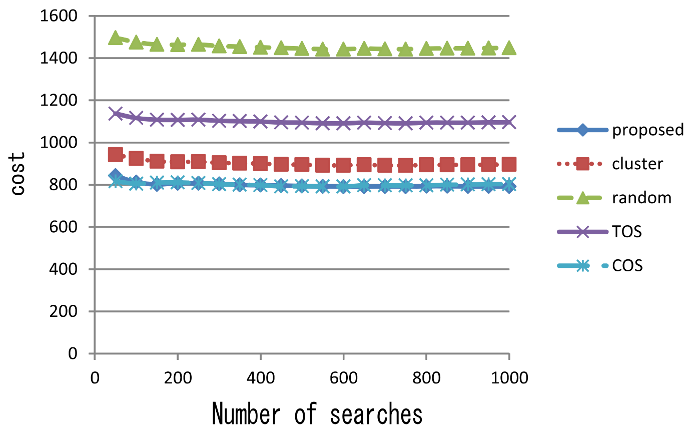

Fig. 6 Relationship between Number of Searches and Average Total Cost (weight ratio = 1:10)

6. References

[1] Google, http://www.google.co.jp

[2] SETI@Home, http://setiathome.berkeley.edu

[3] H. Koide, K. Taniguchi, "A proposal of the Task Scheduling Method for a Grid Computing System Considering Information from the Distributed Memory Management System," Proc. IPSJ Summer Programming Symposium, Sept. 2002.

[4] A. Takefusa, S. Matsuoka, "Performance of a Deadline-Scheduling Scheme on the Computational Grids," IEICE Trans. Info. and Systems, D-I, Vol. J86-D-I, No. 9, pp. 661-670, Sep. 2003.

[5] S. Nishi, H. Yamamoto, T. Ikenaga, Y. Oie, "Network Management Scheme Cooperating with Grid Middleware," IEICE Technical Report, Vol. 104, No. 692, IN2004-333, pp. 441-446, Mar. 2005.

[6] H. Arikawa, H. Sunahara, "A Selection Scheme of the Node to Consider a Load Factor for Campus Grid," IPSJ SIG HPC 91-29, pp. 167-172, Aug. 2002.

[7] S. Kondo, S. Sugawara, Y. Ishibashi, "An Effective Scheduling Scheme for Information Searching with Computational Resources Scattered over a Large-Scale Network," Proc. International Conference on Large-Scale Knowledge Resources (LKR 2008), pp. 351-365, Mar. 2008. (Lecture Notes in Computer Science, Vol. 4938, Springer)

[8] H. Nakamura, S. Sugawara, Y. Ishibashi, "An Effective Searching for Multiple Targets of Information with Shared Computational Resources in a Large-Scale Network," Proc. 2010 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR2010), Jun. 2010.

[9] C. Miura, S. Sugawara, Y. Ishibashi, "A Searching Method for Multiple Target Information with Priority Using Idle Computational Resources Scattered in Networks," IEICE Technical Report, NS2011-88, pp. 35-38, Oct. 2011.

[10] C. Miura, S. Sugawara, Y. Ishibashi, "A Scheduling Method for Multiple Information Searching Taking into Account Usage Efficiency of Idle Computational Resources in Network," IEICE Technical Report, NS2012-160, pp. 111-116, Jan. 2013.

[11] C. Miura, S. Sugawara, "Efficient Scheduling for Concurrent Contents Searching with Idle Computational Resources," BWCCA 2018, LNDECT 25, pp. 1-14, Springer, 2018.