Caching Algorithm for Content-Oriented Networks Using Prediction of Popularity of Contents

Hiroki Nakayama[†], Shingo Ata[‡], Ikuo Oka[‡]
[†] BOSCO Technologies Inc.
1–4–12 Nishishinbashi, Minato-ku, Tokyo 105–0003, Japan
Email:nakayama@bosco-tech.com
[‡] Graduate School of Engineering, Osaka City University
3–3–138 Sugimoto, Sumiyoshi-ku, Osaka 558–8585, Japan
Email:{ata, oka}@info.eng.osaka-cu.ac.jp

***Abstract*—It is important to use cache efficiently for the content deployment in the aspect of reducing load of server and latency, especially in content-oriented networks such as ICN (Information Centric Networking). Since the capacity of cache on each network node is limited, numbers of cache replacement algorithm have been proposed. However, because of previous methods do not consider the rapid fluctuation of content demand, ineffectual cache of content remains which cause a waste usage of cache capacity. In this paper, we propose the method of using the prediction result of content demand. We first demonstrate that by applying prediction can improve the efficiency of cache utilization. However, predicting the demand of whole contents is not realistic in the aspect of computational cost. We therefore propose the method of reducing computational cost greatly without lack of cache efficiency. We demonstrate that by simulating with real monitored data, cache hit rate can be increased about 1.6 times. Furthermore, we show that our method has advantage in many aspects, such as efficient usage of cache and content deployment based on its popularity in the whole network by comparing with other existing methods.**

I. Introduction

It is important to use cache efficiently for the content deployment in the aspect of reducing load of server and latency. Since cache capacity is limited, it is necessary to choose the contents which will be cached based on the caching algorithm. Least Recently Used (LRU) and Least Frequently Used (LFU) are the famous caching algorithm and are used in many situations. However, since these algorithms do not consider the content demand, it does not work effectively and cause a waste usage of cache capacity.

Cache replacement algorithm has been studied extensively in many domains, such as Web caching, Content Delivery Network (CDN) and Information Centric Network (ICN). Especially in ICN, since popularity of content is key factor to increase the performance of cache as [1] demonstrated, number of caching algorithm, which takes content demand into consideration, have been proposed [2]–[4]. Wang et al. [3] proposed the method of placing contents by solving optimization problem, which maximizes the benefit of cache. In this method, they considered the popularity of content by including it into objective function of their optimization problem. On the other hand, Cho et al. [2] proposed the method which the caching decisions are made at individual routers independently. This method adjusts the number of chunks to be cached by considering the content popularity. Although both method make decision of which chunk of content should be cached based on content popularity, the fluctuations of popularity which depends on time progress is not considered in many articles. For example, when news comes in, people try to find related topics so that the number of requests for related contents significantly increases. After a couple of days, however, when people are no longer interested in old news, the frequency of references suddenly decreases. Such fluctuation of content popularity must be taken into consideration to accurately analyze individual trends. Moreover, although the access frequency of content follows Zipf's law, this has not been taken into consideration for fluctuations in time series data in many articles.

We aimed to increase the performance of cache by predicting content popularity for this reason. Especially, in this paper, we focus on improving cache performance of ICN. We first evaluate the effect of predicting content demand and show that there is a big advantage on using prediction as Famaey et al. [4] showed.

However, predicting the popularity for individual content is not realistic in the aspect of computational costs. For example, even if the prediction of popularity of individual content will complete in millisecond order, it takes more than 10 minutes order to predict millions of contents. In fact, the number of contents in the network exceeds 2 billion and it continues increasing day by day [5]. Furthermore, the prediction methods including Auto Regressive (AR) model and neural network needs the time series data of content demand to make prediction. This means that not only CPU power but also large volume of RAM is also required to predict popularity for whole content. Although this point is important for implementing the algorithm, it is ignored completely in [4]. For this reason, we aim to reduce the computational cost by reducing the number of target content for prediction without lacking the performance of caching algorithm which uses content demand prediction.

As the possible method for reducing computational costs, we can consider a method of selecting the popular content in a certain time window for prediction target. However, we have to note that the popularity of content fluctuates with the time progress, which means that we have to keep on changing the prediction target according to the fluctuation of content demand. Therefore, we propose the method of removing the prediction target rapidly accordance with the decrease of content demand. Although we consider of using this algorithm for cache of ICN, our algorithm can be operated in any other system such as CDN.

The remainder of this paper is organized as follows. We first introduce the caching algorithm based demand prediction and demonstrate its efficiency in Section II. We then propose a method of reducing computational costs of prediction by selecting the prediction target in Section III. The results of caching algorithm compared with other existing methods are shown in Section IV. We finally conclude the paper with future topics of research.

II. Caching Algorithm Using Content Demand Prediction

We first introduce caching algorithm using content demand prediction in this section. We use AR model, the method used for predicting time series, to predict content popularity. AR model can be written as

$$x_t = \sum_{i=1}^{p} \alpha_i x_{t-i} + \epsilon_t \tag{1}$$

where x_t is the time series data of content popularity, α_i is the AR coefficient, ϵ_t is the residual, and p (where $p \in \mathbf{R}$) is the order of parameter. According to our previous work, we use Burg [6] method to estimate α_i, and use $p = 5$ for every evaluation in this paper [7]. Fig. 1 indicates the usage of prediction value. As figure shows, prediction of popularity of individual content will be conducted for each prediction interval $\Delta\tau$ using the observed data. Predicted value will be used to compare the popularity of individual content on each cache node to cache the content with higher popularity. In this paper, we call this caching algorithm *prediction cache*.

Here we simply evaluate the efficiency of *prediction cache* with the simulation environment which will be introduced in Subsection IV-A. The results are plotted in Fig. 2(a). The horizontal axis corresponds to time and the vertical axis corresponds to cache hit rate. The evaluation was conducted for 1 week. These results indicate that the *prediction cache* has a clear advantage on cache hit rate comparing with LRU. Using prediction allows us to increase the performance of cache for this reason. However, predicting the popularity of whole content requires tremendous computational cost. This means that in the worst case, the Δt exceeds $\Delta\tau$ depending on the number of content to predict in Fig. 1. Even if we could compute whole popularity of content, there is no sufficient advantage when we take computational cost of *prediction cache* into consideration which is not considered in [4] at all. For this reason, we propose the method of reducing prediction target without lack of performance of *prediction cache*.

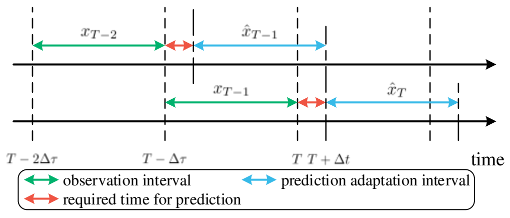
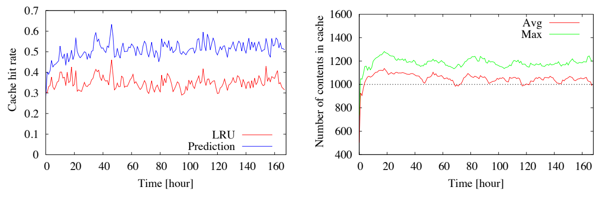


Fig. 1. Usage of prediction value



(a) Cache hit rate (b) Number of contents

Fig. 2. Performance of prediction cache

III. Prediction Target Selection Algorithm

We propose the method of reducing computational cost of *prediction cache* by limiting the target of prediction without lack of performance of prediction cache in this section. Since this algorithm selects the target to predict, we call this method *selection algorithm* in this paper.

A. Limiting the Prediction Target

We first evaluate the necessary targets to maintain the performance of *prediction cache*. Since the number of necessary targets corresponds to the number of content type which have been cached at least once in its timeslot $\Delta\tau$, we here evaluate it first for each cache node in the network. The results are plotted in Fig. 2(b), where the horizontal axis corresponds to time and the vertical axis corresponds to the number of content type which have been cached at least once in its timeslot. We evaluated the average and maximum types of content for each cache node. The additional line plotted with dashed line shows the number of average content cache capacity which is calculated with average cache capacity divided by average content size. It can be considered that limiting prediction target to 1.3 times of cacheable number of contents does not lack performance of *prediction cache*. However by considering the rapid fluctuations of content popularity, we simply limit the prediction target to twice of average content cache capacity.

We then propose the method of selecting the prediction target using 2 observation states for individual content. First we define 2 observation states as follows.

- **Prediction Target (PT) :** Contents included in this observation state will be subject to be predicted and cached. The access frequency of individual content will be contained with time series data to conduct prediction.
- **Selection Target (ST) :** The rest of all contents which is not included in PT belongs to this observation state.

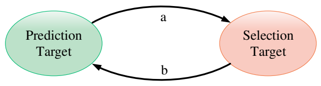
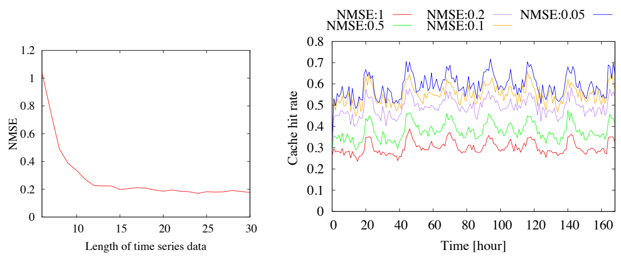


Fig. 3. Prediction target limitation with 2 observation states



(a) Relation of length of time series and prediction accuracy

(b) Effect against cache hit rate

Fig. 4. Effect of prediction error

We will use LFU to select the content which will become popular. LFU will be refreshed when the state transition occurs.

Fig. 3 summarizes observation states and transitions of each state. State transition will be conducted when the new prediction value have been calculated. The content, which belongs to PT, will be transit to ST when its content has not been cached for specific time (Transition a). We call this value, time out value in this paper and use 10 prediction time windows as its value in every case. On the other hand, frequently accessed contents based on LFU included in ST will be transit to PT (Transit b). The number of contents which transits at Transit b depends on the room of PT. Generally, determination of the content which will be cached will conducted in two phases, (1) determine to cache the arriving content or not and (2) determine the content which will be removed from cache. In our proposed method, Phase 1 will be conducted with *selection algorithm* and Phase 2 will be conducted with *prediction cache*. Computational cost for *prediction cache* will be reduced greatly by using this *selection algorithm* since the computational cost depends on the number of contents to predict.

B. Consideration for the Effect of Prediction Error

The computational cost will be reduced greatly by limiting the contents to predict. However, since the popularity of the content which is transited from ST to PT does not have enough length of time series, it is difficult to predict accurate. This is because of the insufficient length of time series data which have been used for estimating parameter for prediction model. Fig. 4(a) indicates this feature for the prediction method use in the previous evaluation. The horizontal axis corresponds to the length of time series data used for estimating parameter of prediction method and the vertical axis corresponds to Normalized Mean Squared Error (NMSE), the metric to evaluate prediction accuracy, respectively. NMSE is defined bellow.

$$\mathrm{NMSE} = \frac{\sum_{i=1}^{T}(x_i - \hat{x}_i)^2}{\sum_{i=1}^{T} x_i^2} \tag{2}$$

This result indicates that actually the insufficient length of time series data used for parameter estimation causes a lack of prediction accuracy.

We also evaluated how such lacks of prediction accuracy affect the performance of *prediction cache*. Based on Fig. 4(a), we found that the prediction error of contents popularity follows a normal distribution with mean 0 and standard deviation σ. Therefore we demonstrated the performance of *prediction cache* where the popularity of individual contents are calculated with the sum of its actual popularity and artificially generated prediction error which follows a normal distribution. Results are shown in Fig. 4(b). The horizontal axis corresponds to time and the vertical axis corresponds to cache hit rate. This result indicates that the performance of cache will decrease depending on the accuracy of prediction. Especially, in the case of NMSE bigger than 0.5 shows the bad cache hit rate. To reduce the prediction target without lack of performance of prediction, it is important to prepare sufficient length of time series of content demand for this reason. Simply we can consider of preparing sufficient length of time series of popularity against whole contents, it is not realistic to hold such a data in individual cache node. Therefore we propose the method of limiting prediction target with 3 observation states by adding another observation state to previous one described on Fig. 3. Fig. 5 summarizes observation states and transitions of each state. Unlike ST, the added observation state, Candidate Target (CT), observes the access frequency of content included in its target and keeps as time series data. The number of targets in CT is same to that of ST. The content belonging to PT transits to ST when it's content have not been cached for specific time (Transit A). To fill the vacancy in PT which occurred with Transit A, content belonging to CT will be transit to PT (Transit B). The content that is transited will be decided based on the average and length of time series of its popularity. Similarly to Transit A, the content which did not transit to PT in CT will be removed from CT (Transit C). We use time out value for Transit C same as that of Transit A. At last, the room made by Transits B and C in CT will be filled with content in ST which is selected based on LFU (Transit D). By using this algorithm it will be able to handle the problem of using insufficient length of time series data for prediction.

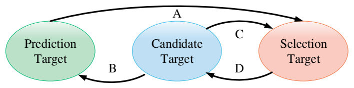


Fig. 5. Prediction target limitation with 3 observation state

IV. Performance Evaluation

We evaluate the performance of our proposed method using our measurement data based ICN simulator in this section. We

TABLE I
SIMULATION ENVIRONMENT

Parameter	Value	Unit
Requirement frequency	Avg 1	Hz
Amount of content	1,000,000	
Size of contents	Avg 10	MB
Size of chunk	0.1	MB
Cache capacity	10,000	MB
Number of cache node	46	
Prediction time unit (Δτ)	10	Min

conduct several comparison evaluations to observe the merits of *prediction cache* and *selection algorithm*.

A. Simulation Environment

In this paper, we prepared chunk based ICN simulator based on ccnSim [8] to evaluate the performance of our proposed method. The parameter of our simulation is listed in Table I which is introduced as YouTube-like environment in [8], where the distribution of content size follows geometry distribution. Each client requires content averaged once per second where its distribution follows real measured data which we introduce in Subsection IV-B. We also use real network topology including 46 cache nodes which is publicly available through Rocketfuel [9]. The simulation is conducted using the traced data of one month with an computer consist of Intel i7 3.40 GHz processor and 8GB RAM.

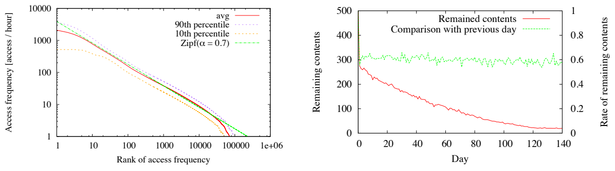
B. Dataset

We used the real measured data to evaluate the performance of our proposed method in this paper. We collected keywords included in query messages in Gnutella, one of the most well-known P2P file-sharing systems. We developed a crawler program by modifying an open source Gnutella client called Phex [10] to collect Gnutella queries. The measurements were conducted from 26 April 2012 to 12 September 2012. During the 20 weeks of the measurements, we collected more than 30 millions of keywords. Especially we focus on one million keywords which point unique content as the request of one's content in our evaluation. Fig. 6(a) indicates the access frequency of our dataset, where the horizontal axis corresponds to rank of access frequency and the vertical axis corresponds to access frequency. Since the requirement of the content fluctuates with progress, Fig. 6(a) shows the distribution averaged with time and its 90th and 10th percentile. The popularity of content closely follows Zipf's law with parameter $\alpha = 0.7$ and $N = 10^6$, where the access frequency of k^{th} popular content is defined by

$$f(k) = \frac{1}{k^{\alpha} \sum_{n=1}^{N} \left(\frac{1}{n^{\alpha}}\right)}. \qquad (3)$$

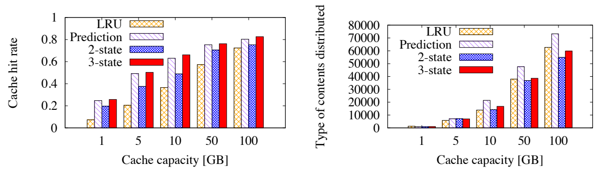
The popularity of contents of YouTube are reported to follow the Zipf's law of $\alpha = [0.6, 0.8]$ [11], [12] which proves that our measured data is useful to consider the real content demand.

We also investigated how the requirements of content fluctuate by focusing on the top 500 content at specific time. The



(a) Distribution of popularity (b) Fluctuation of popularity

Fig. 6. Distribution and fluctuation of contents popularity



(a) Cache hit rate (b) Content cached in the network

Fig. 7. Performance of prediction cache and selection algorithm

results are presented on Fig. 6(b). The horizontal axis shows the time and the vertical axes show the content remaining in the top 500 and ratio of content remaining in the top 500 comparing with previous time slot. The result indicate that the popularity of content decreases with the progress of time. Especially, almost 65% of content will be relegated from the top 500 content after a month has been passed. Moreover, although the ratio of comparison with previous time slot maintains about 0.6, the contents remaining in the top 500 does not decrease geometrically. This means that the contents which are promoted top 500 tend to be relegated easily.

C. Advantage of Using Prediction and Selection Algorithm

We first compare the performance of *prediction cache* and that of LRU. We here use cache hit rate and the number of content type cached in the network for metric. Since the performance of cache algorithm strongly depends on the cache capacity of each cache nodes, we will conduct our evaluation with its parameter of 1GB, 5GB, 10GB, 50GB, and 100GB respectively. The results are presented with histogram in Fig. 7. Fig. 7(a) indicates that cache hit rate will increase by predicting popularity of content. Especially it becomes more advantageous under the situation of small cache capacity. To be more specific, when the cache capacity is 1GB, the cache hit rate of *prediction cache* achieves almost three times than that of LRU. Moreover, as Fig. 7(b) indicates, the number of content type cached in the network of *prediction cache* is also higher than LRU. Predicting content demand is beneficial for this reason.

Next, we compare how the performance of cache prediction will be change by using two kinds of *selection algorithms*. By using 2-state *selection algorithm*, the lack of cache performance can be observed from Fig. 7. On the other hand, it seems that the performance of *prediction cache* does not decrease by using 3-state *selection algorithm* which indicates the usability of CS in the 3-state *selection algorithm*.

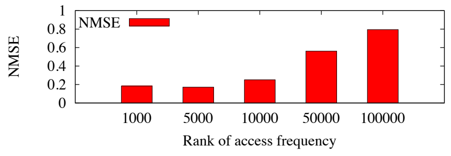
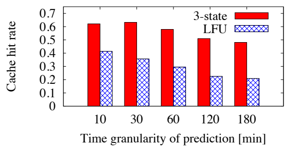
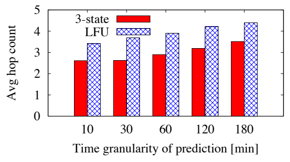


Fig. 8. Prediction accuracy for each rank

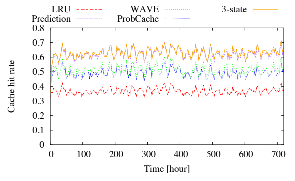


(a) Cache hit rate

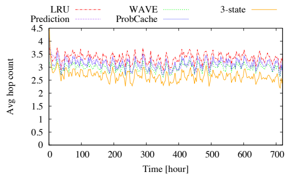


(b) Hop counts

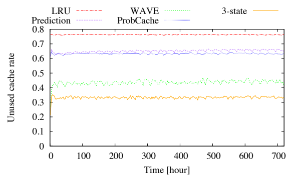
Fig. 9. The influence of prediction window size



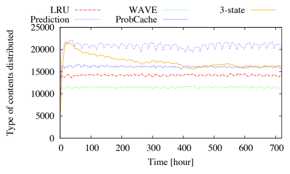
(a) Cache hit rate



(b) Hop counts



(c) Unused cache



(d) Number of content type cached in network

Fig. 10. Comparison with existing methods

Moreover, there is a slight increment on cache hit rate. This advantage is because of ignoring unpopular content as the prediction target which is difficult to predict accurate. We evaluated how the prediction accuracy changes due by the popularity of content to explain our opinion more detail as shown in Fig. 8 with histogram. The result indicates that the more unpopular content is the more inaccurate prediction is. Especially, NMSE of the contents with the rank lower than 10,000 exceeds 0.6. This is because of the time series of unpopular contents tends to include many unexpected features which occurs inaccurate prediction. Inaccurate prediction will cause an inefficient decision in *prediction cache* as Fig. 4(b) indicates. Although the impact of inaccurate prediction against the unpopular content is not so big, cache hit rate will increase by using 3-state *selection algorithm* for this reason.

Also *selection algorithm* has an advantage on feasibility compared against *prediction cache* which is not discussed well in [4]. Although as we discussed about *prediction cache* in Sec. II it is not realistic to predict popularity for each content, *selection algorithm* only predicts against 2,000 contents where whole computations can be concluded in the target time. To be more specific, 2MB of RAM and 500 msec of computational time are enough to handle whole information and conduct computation in the single router. From these results, we can conclude that using prediction can increase the performance of cache, and the performance of *prediction cache* does not decrease by using *selection algorithm* to reduce computational costs.

D. The Influence of Prediction Time Window Size

We here discuss how the performance of *selection algorithm* will changes due to the unit of prediction, which is introduced as $\Delta\tau$. Although we have demonstrated that prediction is useful to increase the performance of cache, it can be considered that its advantage is due to using the content demand for the cache replacement. Therefore we also compared our proposed method with the algorithm with LFU. The results are presented with histogram in Fig. 9. The results indicate that the bigger the unit of prediction the worse cache performance becomes. However, the there is not a big difference between the performance of 10 minutes and that of 30 minutes. This means that AR model can predict the fluctuations for 30 minutes accurate. The results also indicate that using prediction is important to gain performance of cache. This is because of LFU does not consider the rapid fluctuation of content demand which *selection algorithm* does. Not just considering the popularity but also predicting it is important to gain performance for this reason.

E. Comparison Evaluation with Existing Algorithms

Since the efficient usage of cache is a key factor of performance of ICN, several kinds of methods have been proposed. Therefore, we made a comparative evaluation between our proposed method and existing methods. As a comparison target, we prepared ProbCache [13] and WAVE [2]. We used cache hit rate, hop counts to reach the requested content, rate of replaced contents which have not been used at least once, and kinds of contents cached in the network as the metric of performance. The results are presented in Fig. 10.

Figs. 10(a) and 10(b) indicate that using prediction for cache replacement allows providing content faster. Although WAVE also considers the popularity of cache, it cannot achieve big advantage on performance comparing with LRU, since it does not consider the rapid fluctuations of popularity. The algorithm which uses *selection algorithm* and *prediction cache* can achieve 1.3 times of cache hit rate and can reduce hop count to 0.7 times comparing with ProbCache and WAVE. Our proposed method can provide contents faster than other existing methods for this reason.

Figs. 10(c) and 10(d) indicates that there is advantage on efficiency of cache usage too. Especially, Fig. 10(c) indicates that the *selection algorithm* also has an advantage on using cache more efficiently, since *prediction cache* with *selection algorithm* has better performance than WAVE but *prediction cache* without *selection algorithm* does not have. Although the results of Fig. 10(d) indicate that the number of cached content

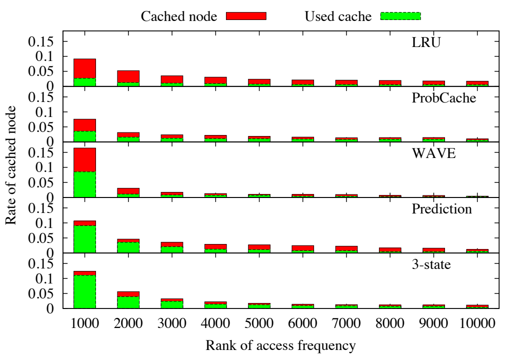


Fig. 11. Rate of cache node caching the target content

types will decrease by using *selection algorithm*, it can cache several kinds of contents comparing with existing methods.

Previous evaluations have only focused on how effectively the cache capacity can be used. However, since cache takes an important role on availability of its contents, which is a great merit of ICN, we will evaluate it by focusing on contents popularity. The results are shown as histogram in Fig. 11. The results indicate the rate of cache node which caches the target content. We first calculated the rate of cached nodes for individual content and after that averaged it for each range of popularity to plot this result. Since cache capacity is finite, we also plotted the rate of nodes which have provided target content to evaluate its efficiency. Therefore there are two goals for this evaluation. One is to distribute cache to multiple node depending on its popularity. We have to note that excessive distribution of popular contents may lose a chance of caching unpopular contents. And another one is to make the rate of cached node to used node (the node which provided target content from its cache) near 1. Based on these ideas, the methods except WAVE achieve the first goal for this evaluation. It seems that there is no problem on caching only popular contents at a glance. However, this means that it will lose a chance of caching unpopular content which lacks the availability of content. Fig. 11 also indicates that the methods which use prediction achieve the second goal more than the methods without prediction, especially in the top 2,000 contents. On the other hand, *selection algorithm* allows making cache usage more effective against the unpopular contents. This is because of ignoring the contents which may occur inaccurate prediction of its popularity. Since the results for contents below the rank of 10,000 are not plotted, there was hardly any difference in each method.

V. Conclusion

We introduced *prediction cache*, the caching algorithm based on prediction of content popularity in this paper. Although we demonstrated that using prediction has a great advantage on increasing performance of cache, the computational cost for predicting the popularity of whole content is not realistic to handle. Therefore we proposed the method of reducing the computational cost for prediction by limiting the prediction target without lack of advantage of prediction. Limiting the target has advantage on not only reducing the computational cost but also increasing performance. We demonstrated that by simulating with real monitored data, cache hit rate can be increased about 1.6 times than that of LRU. Moreover our method has a merit on scalability of delivering content.

We intend to evaluate more detail of our method since there are many aspects which is considered insufficiently, such as timeout parameter, number of limited target, computational costs, and prediction method.

Acknowledgement

This work was partially supported by JSPS KAKENHI Grant Number 25540034.

References

[1] G. Rossini and D. Rossi, "A dive into the caching performance of Content Centric Networking," in *Proceedings of the IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD'12)*, Barcelona, Spain, September 2012, pp. 105–109.

[2] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "WAVE: Popularity-based and Collaborative In-network Caching for Content-Oriented Networks," in *Proceedings of the 1st Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN'12)*, Orlando, Florida, USA, March 2012, pp. 316–321.

[3] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Optimal Cache Allocation for Content-Centric Networking," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP'13)*, Gottingen, Germany, October 2013.

[4] J. Famaey, T. Wauters, and F. D. Turck, "On the Merits of Popularity Prediction in Multimedia Content Caching," in *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management (IM'11)*, Dublin, Ireland, May 2011, pp. 17–24.

[5] G. Gursun, M. Crovella, and I. Matta, "Describing and Forecasting Video Access Patterns," in *Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM'12)*, Orlando, Florida, USA, March 2012, pp. 460–468.

[6] J. P. Burg, "Maximum Entropy Spectral Analysis," http://sepwww.stanford.edu/theses/sep06/.

[7] H. Nakayama, S. Ata, and I. Oka, "On Comparison of the Efficiency of Auto Regressive Estimation Methods for Predicting Traffic Trend Patterns," in *Proceedings of the 14th Asia-Pacific Network Operations and Management Symposium (APNOMS'12)*, Seoul, South Korea, September 2012.

[8] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," *Relatorio tecnico, Telecom ParisTech*, 2011.

[9] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proceedings of the ACM SIGCOMM 2002*, Pittsburgh, PA, USA, August 2002, pp. 133–145.

[10] "Phex," http://www.phex.org/mambo/.

[11] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System," in *Proceedings of the ACM Internet Measurement Conference 2007 (IMC'07)*, New York, NY, USA, August 2007, pp. 1–14.

[12] X. Cheng, C. Dale, and J. Liu, "Statistics and Social Network of YouTube Videos," in *Proceedings of the 16th International Workshop on Quality of Service (IWQoS'08)*, Enschede, Netherland, June 2008, pp. 229–238.

[13] G. P. Ioannis Psaras, Wei Koong Chai, "Probabilistic In-Network Caching for Information-Centric Networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking (ICN'12)*, Helsinki, Finland, August 2012, pp. 55–60.