Priority Control Based on Website Categories in Edge Computing

Noriaki Kamiyama*†, Yuusuke Nakano*†, Kohei Shiomoto†,
Go Hasegawa‡, Masayuki Murata*, and Hideo Miyahara*
*Department of Information Science, Osaka University, Osaka 565-0871, Japan
Email: {kamiyama.noriaki, nakano.yuusuke, murata, miyahara}@ist.osaka-u.ac.jp
†NTT Network Technology Labs, Tokyo 180-8585, Japan, Email: shiomoto.kohei@lab.ntt.co.jp
‡Cybermedia Center, Osaka University, Osaka 560-0043, Email: hasegawa@cmc.osaka-u.ac.jp

***Abstract*—Modern websites consist of many rich objects dynamically produced by servers and client terminals at diverse locations. To deliver dynamic objects efficiently, edge computing in which objects are dynamically generated and delivered to client terminals on edge servers located at edge nodes will be effective. It is anticipated that the effectiveness of edge computing depends on the geographical pattern of object deployment, and the tendency of geographical distribution of objects is different among website categories, e.g., Sports and News, so it seems effective to prioritize website categories for edge computing. In this paper, we first propose a platform for measuring the geographical tendency of object deployment in each website category. We then propose to differentiate the caching priority in edge computing among website categories based on the measured deployment pattern of objects. Through the experience of accessing about 1,000 of the most popular websites from 12 locations worldwide using PlanetLab, we clarify that we can improve the reduction ratio of web response time by about 20% by carefully differentiating caching priority among website categories.**

I. Introduction

In recent years, a large portion of Internet traffic has been dominated by HTTP traffic. However, it has been reported that two-thirds of the users encountered slow websites every week, and about half of those users abandoned websites after experiencing performance issues [4]. Furthermore, it has been reported that a 400-millisecond delay resulted in a 0.74% decrease in searches on the Google search engine [18], and that Amazon′s revenue increased by 1% for every 0.1 second reduction in web page load time [19]. Therefore, reducing web response time is urgent for many ISPs and content providers. It is important to adequately control web traffic to improve user-perceived quality and to reduce the amount of network resources consumed.

Each website consists of a large number of data objects, and objects are transmitted from object servers using HTTP. With traditional websites, static objects are stored at servers, and web browsers simply download them. Content delivery networks (CDNs) that use a number of cache servers deployed in multiple networks have been widely used as a method to efficiently transmit web traffic and reduce response time [9][11][17]. Currently, 74% of the 1,000 most frequently accessed websites use CDNs [11]. Such networks are effective for delivering static objects because identical data can be used for multiple users. However, the ratio of dynamic objects generated when executing programs, e.g., Java server pages (JSPs) and Servlet, at servers or executing JavaScript, e.g., Ajax and document object model (DOM), at client terminals, has increased recently [4]. Content delivery networks are not effective for delivering dynamic objects because such objects cannot be reused among multiple users in many cases. The problem of a long time being required to access websites consisting of many dynamic objects has been pointed out [18][19].

To deliver dynamic objects efficiently, edge computing, in which objects are dynamically generated and delivered to client terminals on edge servers located at edge nodes, seems effective [5][14][15]. More recently, similar idea by the name of *fog computing* has been also proposed [3][20][21]. By caching application codes for generating dynamic objects, the same application codes can be reused among various users, so we can expect to avoid the delay caused to obtain the application codes from remote servers. However, because the total number of websites worldwide is enormous and continues to increase rapidly, edge servers cannot support all the objects of the entire set of websites around the world, so it is important to carefully select objects to maximize the effectiveness of edge computing.

The worldwide deployment patterns of objects will strongly affect web response time, and the object-deployment patterns may tend to differ among website categories, e.g., Sports and Business. In our previous work [8], we showed the result of measuring traffic generated when browsing about 1,000 of the most popular websites from 12 locations worldwide and confirmed the difference in the tendencies of object-deployment patterns among 16 website categories. For example, objects of categories with high locality, e.g., Home and Shopping, tend to be obtained from servers near users in each region, whereas those of categories with low locality, e.g., Adult and Society, tend to be obtained from remote servers because they are obtained from an identical and specific location even when accessing from different regions around the world. Therefore, it is anticipated that the effectiveness of edge computing in reducing web response time will depend on the object category and that priority control for the edge-computing target among website categories will be effective.

In this paper, we propose a method of differentiating the caching priority among the website categories in edge computing based on the geographical pattern of object deployment measured from the edge servers. The contributions of this paper are summarized as follows:

- To effectively reduce the web response time with dynamic objects, we propose a platform for measuring the geographical tendency of object deployment in each website category, and we propose to dynamically configure the caching priority among website categories in edge computing based on the measured deployment pattern of objects.
- To clarify the potential and effectiveness of the proposed caching method in edge computing, we roughly analyze the lower bound of the reduction effect of response time obtained by differentiating the priority of caching in edge computing among website categories.

In Section II, we summarized the measurement procedure and the results of object-deployment tendencies in each website category by accessing websites from various access points using PlanetLab [8]. In Section III, we discuss our platform for configuring the caching priority in edge computing based on the active measurement from edge servers. In Section IV, we evaluate the effectiveness of the proposed caching method in edge computing and conclude the manuscript in Section V.

II. Analysis of Web-Traffic Patterns

In this section, to clarify the possibility and significance of differentiating priorities among website categories in edge computing, we summarize the results of analyzing the tendencies of object deployment of each website category by actively measuring the traffic patterns generated when accessing various websites from various access points [8].

A. Measurement Procedure

First, we briefly describe the procedure used to measure the traffic properties when accessing various websites.

1) Acquisition of Web-Traffic Properties: The traffic generated when users access popular websites should be analyzed to investigate trends in the communication patterns of websites. Quantcast provides a ranking list of websites accessed by users in each country [13]. We used this ranking list for selecting the most popular websites. The URLs shown in this access ranking list are the home pages of each website, so we analyzed the properties of traffic generated only when accessing the home pages. We did not evaluate which pages can be accessed from the home pages; therefore, we did not evaluate user behavior when they were browsing websites. However, we were able to roughly investigate web traffic trends by analyzing the communication properties generated when users access many websites. We classified the selected URLs into website categories to investigate the differences in communication patterns among various types of websites. The website of Alexa provides URL lists classified by category, e.g., Arts and Business [1], and we classified the extracted data into the 16 website categories shown in Table I on the basis of this list.

We acquired HTTP archive record (HAR) files [10] to obtain the communication properties generated when sending a GET message of an HTTP request from the probing terminal. In the HAR files, various communication properties, e.g., the host URL from which each object is downloaded, size of each object, and delay caused in obtaining each object, are output as JavaScript Object Notation (JSON). We continuously and automatically accessed a large number of websites by using the netsniff.js executable on phantomjs in which JavaScript can be executed on the command line [6]. Many cacheable objects, e.g., video or images, are cached at client terminals, and these objects in the local cache are reused when accessing the same websites from the same client terminal. We obtained all the objects from remote servers by invalidating the local cache of the probing client terminal. After obtaining the HAR file for each website accessed, we extracted various data from the information of each object included in each obtained HAR file.

TABLE I
Number of websites for each website category used in clustering analysis

ID	Category	ID	Category
C1	Business	C9	Home
C2	Computers	C10	Shopping
C3	News	C11	Adult
C4	Reference	C12	Arts
C5	Regional	C13	Games
C6	Science	C14	Kids & teens
C7	Society	C15	Recreation
C8	Health	C16	Sports

TABLE II
Measurement locations

ID	Area	Location	ID	Area	Location
L1	NA	Massachusetts	L7	OA	Australia
L2	NA	Wisconsin	L8	OA	New Zealand
L3	NA	California	L9	AS	Japan
L4	EU	Ireland	L10	SA	Ecuador
L5	EU	Germany	L11	SA	Argentina
L6	RU	Russia	L12	AF	Reunion

We accessed websites from multiple access locations using PlanetLab [12], which is an overlay network constructed on the Internet and consists of over 500 hosts worldwide. Using PlanetLab, we were able to execute various kinds of programs on a number of selected hosts. By executing the procedure described in the previous two subsections, we were able to access various websites from various locations worldwide. The probing terminals we used were various hosts on PlanetLab. We also measured the RTT from each access host to each object server in addition to the statistical data obtained from the HAR files. To measure the RTT of each object, we automatically sent a ping command to each object server immediately after obtaining the HAR file of each website at each PlanetLab access host.

We selected a total of 12 measurement locations on PlanetLab: three points in North America (NA), two in Europe (EU), one in Russia (RU), two in Oceania (OA), one in Asia (AS), two in South America (SA), and one in Africa (AF). These 12 locations are shown in Table II. We selected 300 websites with the highest access count from each of the 16 website categories. From each location, we continuously accessed these websites starting at midnight (0:00) and noon (12:00) local time of each access location. The total number of websites from which the HAR files were successfully obtained was 1,124 at midnight and 927 at noon.

By investigating the tendencies in web traffic of objects delivered using CDNs (denoted as *CDN objects*) and objects delivered without using CDNs (denoted as *non-CDN objects*), we can investigate the geographical tendency of cache deployment of CDNs and the locations where original objects of websites are provided. We classified the objects extracted from the HAR files into two sets, i.e., CDN objects and non-CDN objects, by creating a list of second-level domains of hosts delivering CDN objects.

First, we listed the second-level domains of various CDN providers by manually checking websites of various CDN providers. We obtained a total of 44 second-level domain names of CDN caches, e.g., edgesuite.net, cloudfront.net, and akamaiedge.net. The domain names of objects extracted from the HAR files are those of content providers, e.g., www.yahoo.com, and the domain names of hosts delivering objects, e.g., host1.akamaiedge.net, differ from those extracted from the HAR files. We obtained the domain names of the hosts actually delivering objects by using the dig command. Finally, we identified CDN objects by comparing the second-level domain obtained by the dig command with each second-level domain included in the generated list.

B. Clustering Analysis of Web Traffic Properties

To understand the manner in which properties tend to differ when accessing websites from various locations, we used clustering analysis. We accessed various websites, Y_1 and Y_2, from various measurement locations, X_1, X_2, and X_3, at each measurement time t. When we accessed each website from N access locations, we obtained N results of each property, e.g., average RTT, for the same website. Therefore, we can construct $\boldsymbol{v}(y,t)$, a vector of N dimensions in which each element $v_{y,t,k}$ ($1 \leq k \leq N$) is the value v measured at location k when accessing website y at t. When we accessed M websites at t from N locations, we obtained M vectors $\boldsymbol{v}(y,t)$ for $1 \leq y \leq M$. Using the obtained M vector $\boldsymbol{v}(y,t)$, we applied the clustering analysis to investigate the trends in the way each v differs among the accessed locations.

The k-means method is the most widely used method for centroid-based clustering, and we applied it to cluster websites on the basis of $\boldsymbol{v}(y,t)$. It is widely known that k-means method results are strongly affected by the initial clusters, i.e., the initial k centroids. One of the major problems with the k-means method is that the approximations found can be arbitrarily bad with respect to the objective function compared to the optimal clustering. To address this problem, Arthur et al. proposed the k-means++ algorithm, which is a procedure to initialize the cluster centers before proceeding with the standard k-means optimization iterations [2]. The basic idea of this method is spreading out the k initial cluster centers. The first cluster center is chosen uniformly at random from the data points that are being clustered, after which each subsequent cluster center is chosen from the remaining data points with probability proportional to its squared distance from the points closest to the existing cluster center. We use the k-means++ method to avoid the initial cluster problem. Moreover, the k-means method results also strongly depend on the parameter k, i.e., the number of clusters. In this study, we used the JD method proposed by Jain et al. to optimally determine k [7].

C. Geographical Distribution of Original Objects

We investigated the tendency in the geographical distribution of original objects in each URL category through the clustering analysis of RTT for non-CDN objects. Figures 1(a) and (b) respectively plot the centroids of the average RTT of non-CDN objects of websites classified into each cluster and the ratio of websites classified into each cluster when using the midnight dataset. We classified the websites into four clusters and observed the different tendencies among the URL categories in average RTT, unlike the case of average distance.

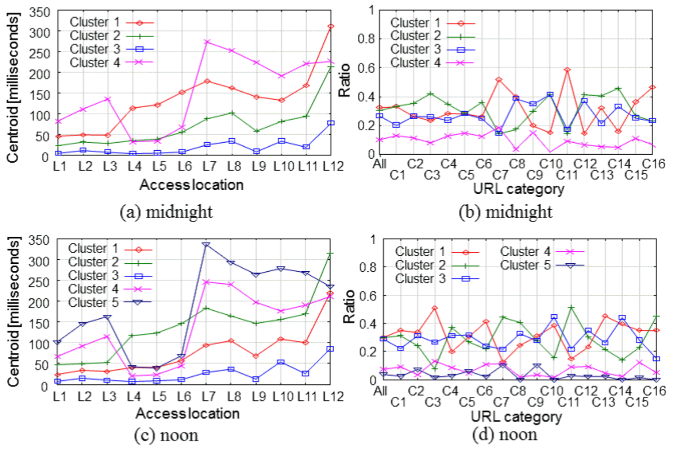

Fig. 1. (a)(c) Centroids of average RTT of non-CDN objects at each access location, (b)(d) ratio of websites classified into each cluster in each website category

The average RTT to servers providing original objects of websites classified into Cluster 1 was small only in North America, and the ratio of Society, Adult, Recreation, and Sports websites classified into Cluster 1 was large. The websites of these categories tended to be provided by content providers in North America. The centroid of Cluster 2 was small in North America, Europe, and Asia, and more websites for Computers, News, Reference, Science, Arts, Games, and Kids & Teens were classified into this cluster. Many content items of these categories were provided by content providers in North America, Europe, and Asia. For example, we can assume that many websites of Games and Kids & Teens were provided by major Japanese content providers. We can say that the geographical locality of many websites classified into Clusters 1 and 2 was weak, and identical content tended to be viewed from various regions.

On the other hand, the centroid of Cluster 3 was small in all the areas excluding Africa, and more websites of Home and Shopping tended to be classified into this cluster. The geographical locality of many websites of Home and Shopping was high, and the websites of these categories tended to be provided from various countries. Therefore, original objects were obtained from servers provided at locations close to each access location. Finally, the average RTT to servers providing original objects of websites classified into Cluster 4 was small only in Europe, and only less than 10% of websites of all categories were classified into this cluster.

Figures 1(c) and (d) respectively plot the same properties using the noon dataset in which the websites were classified into five clusters. Clusters 1, 2, and 3 in the noon dataset correspond to Clusters 2, 1, and 3 in the midnight dataset, respectively. Moreover, Clusters 4 and 5 in the noon dataset correspond to Cluster 4 in the midnight dataset. We confirmed that the tendency of the centroids of clusters was similar to that in the midnight dataset.

III. Differentiating Priority among Website Categories in Edge Computing

In Section II-C, we confirmed that the tendencies of geographical distribution of original objects are different among website categories, and we can roughly classify the website into two broad groups: universal, e.g., Adult and Society, and localized, e.g., Home and Shopping. On the basis of this finding, the following approach would seem to be effective for priority control of edge servers:

Servers providing objects of websites of the universal group with low locality tend to concentrate in North America, so an effective way to improve the user response time of websites would be to preferentially deliver the objects of these categories from edge servers over various areas in addition to North America.

However, the object deployment pattern continues to change dynamically over weeks or months, and the priority of website categories in edge computing should be dynamically adjusted. To achieve this goal, the network operator needs to periodically collect the data of the web communication pattern measured at various points in the network. Therefore, first we propose a platform for measuring the RTT to object servers of various websites from the edge servers and analyzing the geographical tendency of object deployment at the controller to effectively configure the caching priority among website categories in edge computing.

A. Platform for Measuring Deployment of Web Objects

To grasp the geographical tendencies of web-object deployment, the network operator needs to measure the RTT from various access points on the network. Here, we propose a measurement platform in which all the edge servers act as the measurement hosts and execute the active-measurement procedure described in Section II-A. As shown in Fig. 2, the proposed platform consists of the controller of the network operator, edge servers, and webservers of various websites. At any time, the network operator can freely update the probing configuration at the controller and send the probing configuration to edge servers to update their probing configurations to the latest one set by the network operator, as shown in Fig. 2(a). As a result, the probing configuration set by the network operator is dynamically reflected to all the edge servers. The probing configuration includes the list of URLs each edge server accesses and the time when starting the active measurement of web-traffic properties.

On the basis of the probing configuration, each edge server sequentially accesses multiple websites according to the URL list set by the probing configuration, as shown in Fig. 2(b), and measures the RTT to web servers providing each object of websites according to the procedure mentioned in Section II-A. Then each edge server sends the obtained RTT data to the network operator, as shown in Fig. 2(c). The network operator derives the geographical tendencies of the RTT by means of clustering analysis, as described in Section II-B. On the basis of the obtained tendencies, the network operator sets the priority among website categories for edge computing and informs the determined category priority to edge servers, as shown in Fig. 2(d). For example, the network operator can set high priority to website categories that are classified into the *universal* group with the RTT at many areas much larger than that at the specific area, e.g., North America.

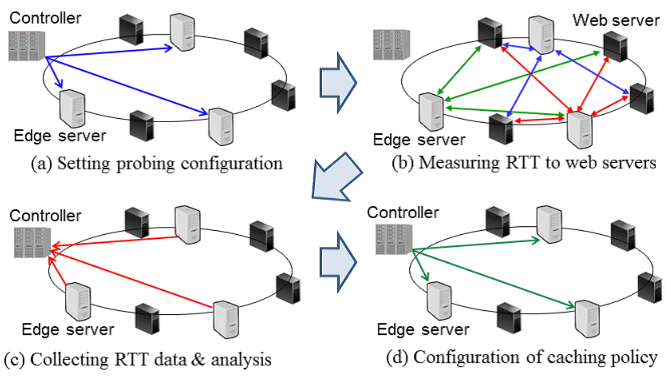
Controller
Edge server
(a) Setting probing configuration
Web server
Edge server
(b) Measuring RTT to web servers
Controller
Edge server
(c) Collecting RTT data & analysis
Controller
Edge server
(d) Configuration of caching policy

Fig. 2. Platform for measuring geographical deployment of web objects

B. Priority Control among Categories in Edge Computing

On the basis of the tendencies of object deployment, which are analyzed at the controller of the network operator, the network operator sets the priority among website categories for edge computing and informs the edge servers of the determined category priority. Based on this priority, each edge server autonomously executes the priority control among website categories. For selecting the objects cached at edge servers, we can apply the same procedure with that currently used in many CDNs.

For domain name system (DNS) queries issued by user terminals when accessing websites, DNS servers of the ISPs operating the edge computing platform select one edge server and forward the user query to the selected edge server. The edge server selected by the DNS generates an object and delivers it to the requesting user if the application code of the requested objects exists in its memory, i.e., *cache hit*. Otherwise, i.e., *cache miss*, the edge server obtains application codes from the original server, dynamically generates objects, and delivers them to the requesting user. At this time, the edge server determines whether to store the obtained application code in the memory of the edge server based on the website categories. For example, the edge server can store the application code only if the website category of the obtained application code is in the high priority class, i.e., universal group. If the unused memory capacity is not sufficient to store the obtained application codes, some application codes in memory are removed using least recently used (LRU).

IV. Reduction Effect of Web Response Time

To show the potential of the proposed caching method of differentiating caching priority among website categories in edge computing, we roughly evaluated the response time when objects of the selected website categories are provided from the edge servers, based on the main findings of the experiments discussed in Section II-C.

A. *Rough Estimate of Reduction Effect of Web Response Time*

When objects are delivered from the edge servers, we can roughly say that the RTT to the remote web servers from the client terminals is subtracted from the total delay of objects. Let D_x and F_x denote the time reduced by delivering objects of website x from the edge servers and the response time when browsing x without using edge servers, respectively. Moreover, we define G_x, the reduction ratio of the response time of x, by $G_x = D_x/F_x$. We also define $\boldsymbol{S}_x$ as the set of web servers providing objects of x, M_s, as the number of objects of x provided by web server s, and R_s as the average RTT from the edge server and s. By delivering an object whose origin server is s from the edge server to the user terminal, we can expect to reduce R_s from the total response time. Moreover, one RTT is necessary for setting up one TCP connection between the user terminal and each web server. Therefore, if the web browser sequentially downloads all the objects, we have $D_x = \sum_{s \in \boldsymbol{S}_x} \sum_{s=1}^{M_s+1} R_s$.

However, many web browsers support the function of simultaneously downloading multiple objects in parallel to reduce the web response time, so the reduction in web response time by edge computing is much smaller. The maximum number of parallel sessions established with one web server is limited to below the upper limit P to avoid congestion at web servers. Although most browsers download only two objects in parallel, newer browsers open more than two connections, e.g., Internet Explorer 8 (six), Firefox 3 (six), Safari 3 (four), and Opera 9 (four) [16]. Moreover, when the cache hit ratio at the edge servers is given by H, we assume that objects randomly selected with the probability of H are delivered from the edge servers. Therefore, in an ideal case, the web browser simultaneously starts downloading objects from all web servers and downloading up to P objects from each web server. In this ideal case, we have

$$D_x = \max_{s \in \boldsymbol{S}_x} \left\{ \left\lceil \frac{M_s H}{P} \right\rceil + \lfloor H \rfloor \right\} R_s. \tag{1}$$

We note that the second term, $\lfloor H \rfloor$, takes unity when $H = 1$ or zero when $H < 1$, and this term corresponds to one RTT required for setting up one TCP connection between the user terminal and the web server. In reality, web browsers cannot start downloading objects until the completion of obtaining other objects due to various reasons, e.g., maintaining the consistency of DOM, so web browsers cannot start downloading all the objects at the same time [19]. Therefore, due to the dependencies among objects, D_x will be much larger than that obtained by (1) in many cases, and (1) gives the lower bound of D_x. When deriving D_x from (1), we applied the measured value obtained in the experiment mentioned in Section II to $\boldsymbol{S}_x$, M_s, and R_s, whereas we gave P and H as the setting parameters.

B. *Reduction in Response Time in each Website Category*

First, we compared the effectiveness of edge computing on reducing the response time among the website categories. As an example, we evaluated D_x and G_x among four website categories: Adult, Society, Home, and Shopping. As mentioned in Section II-C, we can divide these four categories into two groups: *universal* and *localized*. The weak-locality ones Adult and Society were classified into the *universal* group, and the strong-locality categories Home and Shopping were classified into the *localized* group.

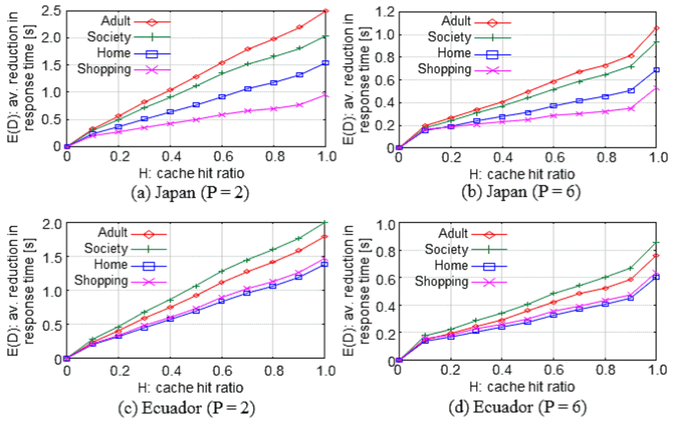

Fig. 3. Average reduction in response time in each website category against cache hit ratio

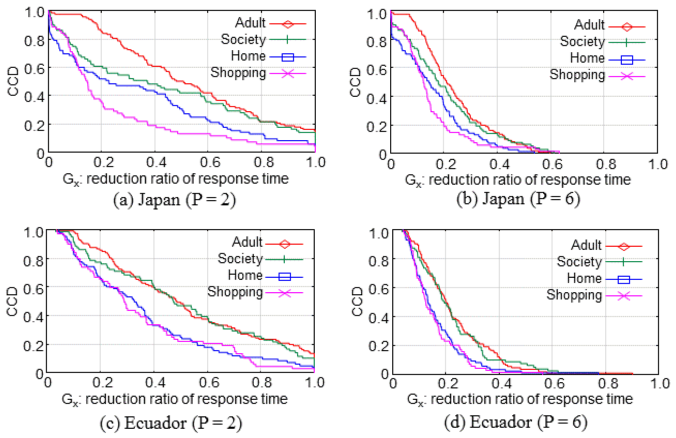

Fig. 4. Complementary cumulative distribution (CCD) of reduction ratio of response time obtained from edge computing

Figures 3(a) and 3(b) plot $E(D)$, the average D_x of websites, in each of the four categories, Adult, Society, Home, and Shopping, against H, the cache hit ratio, based on $\boldsymbol{S}_x$, M_s, and R_s measured at the PlanetLab host in Japan when setting $P = 2$ and 6, respectively. Figures 3(c) and 3(d) also show $E(D)$ based on the data measured at the PlanetLab host in Ecuador. We observed that $E(D)$ of universal categories, Adult and Society, was much larger than those of localized categories, Home and Shopping, in the wide range of H. When $P = 2$ and $H = 1$ in Japan, for example, the response time of the universal websites was reduced by more than two seconds on average, whereas that of the localized websites was reduced by less than 1.5 seconds on average by delivering objects from the edge servers.

Figures 4(a) and 4(b) show the complementary cumulative distribution (CCD) of G_x, the reduction ratio of response time of websites in each of the four categories measured in Japan when $H = 1$ and setting $P = 2$ and 6, respectively. Figures 4(c) and 4(d) also plot the CCD of G_x based on the data measured in Ecuador. We also confirmed the difference in the reduction in response time between the universal and localized websites.

C. Differentiating Website Categories on Edge Computing

To investigate the effect of differentiating the caching priority among website categories in edge computing, we compared $E(G)$, the average reduction ratio of response time G_x, among the following three cases determining how the four categories, Adult, Society, Home, and Shopping, were treated.

- **Without priority differentiation**: We equally treated all the four website categories without any differentiation and assumed that objects of half the websites for each category are delivered from the edge server. Therefore, $E(G)$ was obtained by $E(G) = \sum_{c\in\boldsymbol{C}}\sum_{x\in\boldsymbol{W}_c} G_x/N_c/4$, where $\boldsymbol{C}$ is the set of all the four website categories, $\boldsymbol{W}_c$ is the set of measured websites of category c, and N_c is the number of websites included in $\boldsymbol{W}_c$.
- **Prioritizing Universal group**: We prioritized the universal group websites, i.e., Adult and Society, and assumed that only objects for these categories are delivered from the edge servers. Let $\boldsymbol{C}_U$ denote the set of universal group websites among the four categories, and we have $E(G) = \sum_{c\in\boldsymbol{C}_U}\sum_{x\in\boldsymbol{W}_c} G_x/N_c/2$.
- **Prioritizing Localized group**: We prioritized the localized group websites, i.e., Home and Shopping, and assumed that only objects for these categories are delivered from the edge servers. We have $E(G) = \sum_{c\in\boldsymbol{C}_L}\sum_{x\in\boldsymbol{W}_c} G_x/N_c/2$, where $\boldsymbol{C}_L$ denotes the set of localized group websites among the four categories.

Figure 5 plots the $E(G)$ for each of the three strategies at each of the 12 locations shown in Tab. II when setting $H = 1$. We confirmed that $E(G)$ can be improved by prioritizing weak-locality universal group websites in the edge computing, and that the effect of prioritizing website categories in edge computing was more pronounced outside North America. For example, when $P = 2$ in Australia, we could reduce the average web response time by about 45% when prioritizing universal group websites in the edge computing, whereas the reduction effect of average web response time without differentiating website categories and with prioritizing localized group websites was just about 35% and 25%, respectively. This clarified the potential and effectiveness of differentiating website categories in the priority control of edge computing.

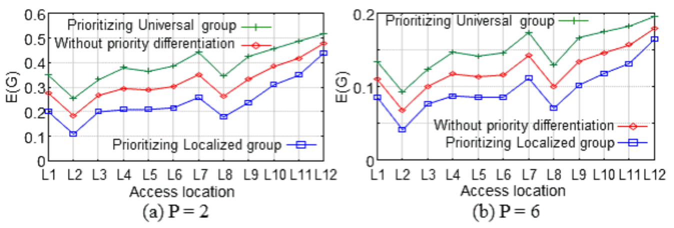

Fig. 5. Average reduction ratio of response time obtained by category differentiation at each location

V. Conclusion

Recently, the communication structure brought about by accessing websites has become more complex. To efficiently deliver dynamic objects, edge computing in which edge servers located close to users cache the application code, dynamically generate objects, and deliver them to users on behalf of remote object servers seems effective. However, the degree to which these techniques are effective strongly depends on object deployment patterns. In this study, we proposed a platform for measuring the geographical deployment of web objects from edge servers and configuring the caching policy based on website categories in edge computing. Through the simple analysis of the reduction in response time, we showed that we can improve the reduction ratio of web response time by about 20% by carefully differentiating caching priority among website categories.

References

[1] Alexa, http://www.alexa.com/topsites/category.

[2] D. Arthur and S. Vassilvitskii, k-means++: the advantages of careful seeding, ACM SODA 2007.

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, Fog computing and its role in the internet of things, ACM MCC 2012.

[4] M. Butkiewicz, H. V. Madhyastha, and V. Sekar, Understanding Website Complexity: Measurements, Metrics, and Implications, ACM IMC 2011.

[5] A. Davis, J. Parikh, and W. E. Weihl, Edgecomputing: extending enterprise applications to the edge of the internet, ACM WWW 2004.

[6] GitHub, Network Monitoring, http://github.com/ariya/phantomjs/wiki/Network-Monitoring

[7] A. L. Jain and R. C. Dubes, Algorithms for Clustering Data, Englewood Cliffs, NJ Prentice-Hall, 1988.

[8] N. Kamiyama, Y. Nakano, K. Shiomoto, G. Hasegawa, M. Murata, and H. Miyahara, Investigating Structure of Modern Web Traffic, IEEE HPSR 2015.

[9] E. Nygren, R. Sitaraman, and J. Sun, The Akamai Network: A Platform for High-Performance Internet Applications, ACM SIGOPS 2010.

[10] J. Odvarko, HAR Viewer, Software is hard, http://www.softwareishard.com/blog/har-viewer.

[11] J. Ott, M. Sanchez, J. Rula, F. Bustamante, Content Delivery and the Natural Evolution of DNS, ACM IMC 2012.

[12] PlanetLab, https://www.planet-lab.org/

[13] Quantcast, http://www.quantcast.com/top-sites-1.

[14] M. Rabinovich, Z. Xiao, and A. Aggarwal, Computing on the Edge: A Platform for Replicating Internet Applications, WCW 2003.

[15] S. Sivasubramanian, G. Pierre, M. Steen, and G. Alonso, Analysis of Caching and Replication Strategies for Web Applications, IEEE Internet Computing, 11(1), pp.60-66, 2007.

[16] S. Souders, High Performance Web Sites: Essential Knowledge for Front-End Engineers, O′Reilly Media, 2007.

[17] A. Su, D. Choffnes, A. Kuzmanovic, and F. Bustamante, Drafting Behind Akamai: Inferring Network Conditions Based on CDN Redirections, ACM Trans. Networking, 17(6), pp. 1752-1765, 2009.

[18] S. Sundaresan, N. Feamster, R. Teixeira, and N. Magharei, Characterizing and Mitigating Web Performance Bottlenecks in Broadband Access Networks, ACM IMC 2013.

[19] X. Wang, A. Balasubramanianm A. Krishnamurthy, and D. Wetherall, Demystifying Page Load Performance with WProf, NSDI 2013.

[20] S. Yi, Z. Hao, Z. Qin, and Q. Li, Fog Computing: Platform and Applications, IEEE HotWeb 2015.

[21] J. Zhu, D. Chan, M. Prabhu, P. Natarajan, H. Hu, and F. Bonomi, Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture, IEEE SOSE 2015.