Priority Access Mechanism for Improving Responsiveness to Users through Cache Server and Priority Access Management Mechanism

Hisanori MIZOBUCHI and Keizo SAISHO
Kagawa University
2217-20 Hayashi-cho, Takamatsu 761-0396, Japan
s15g485@stu.kagawa-u.ac.jp, sai@eng.kagawa-u.ac.jp

ABSTRACT
This research aims at caching non-interactive dynamic contents such as news and weather, and blogs for read only users. We think that there is no problem for users even if they browse these dynamic contents in several seconds later. However, practical use of cached dynamic contents requires quick cache update. We have developed Priority Access Mechanism and Priority Access Management Mechanism. Priority Access Mechanism assigns the quota of the computer resource to handle access from cache servers independently of other accesses. By this mechanism, Web server can respond to cache servers quickly even if there is huge amount of other requests. Priority Access Management Mechanism registers new cache server and client from which accesses to be recognized as priority access. By registering client, users can update their page using Web interface without stress. This paper describes design, implementation and evaluation of Priority Access Mechanism, and Priority Access Management Mechanism. Through experiments, we confirm that it is possible to adjust the quota of the computer resources to process accesses from cache server, cache server can update contents in a short time and respond to users quickly, and cache server and client can be registered dynamically.

KEYWORDS
priority access, cache server, quick response, dynamic contents, dynamic registration

1 INTRODUCTION
Recently the Internet becomes widely used and a lot of people use it. Surge of traffic caused by certain events such as a topic of TV show and disaster increases load of servers and delays response from them. Load distribution technics are generally used to avoid overload of servers. There is a Web system that uses cache servers or mirror servers for load distribution. In this paper, we call such Web system “distributed Web system”. We developed a distributed Web system consisting of Web server and cache servers. The system varies the number of cache servers according to the load[1].

We think that dynamic contents such as news and weather, and blogs for read only users can be cached in short term by setting the expiration date of cache of dynamic contents very short. It can be lighten server load sufficiently with short term caching. However, it cannot use this idea when cache update takes long time. In order to avoid this situation, we divide accesses into two groups, one is priority access group including accesses from cache servers and the other is normal access group including accesses from clients. And then we develop Priority Access Mechanism to process priority access group prior to normal access group. Since the mechanism can serve dynamic contents to cache servers quickly, the quality of cache contents served by cache servers can be improved.

Priority Access Mechanism regards accesses from specific (priority) hosts as priority accesses. By registering cache servers as priority hosts, accesses from cache servers are processed prior to other accesses. Since cache servers are appended or removed according to load dynamically, we also develop Priority Access Management Mechanism which registers appended cache server as priority host to Priority Access Mechanism and deletes registration of removed cache server. By registering client as priority host, users can update their page using Web interface without stress even when Web server stays in overload.

In this paper, design, implementation and evaluation of Priority Access Mechanism and Priority Access Management Mechanism are described.
2 RELATED WORK

There are several researches to classify accesses into some classes and assign them appropriate priority.

Lu, L., Cherkasova, L., de Nitto Personè, V., Mi, N., and Smirni, E. presented “a novel autonomic session based admission control policy”[2]. It uses queue to control the number of active simultaneous processing accesses and the number of maximum acceptance accesses, and reject accesses over maximum acceptance number. It is similar to our system but ours’ does not reject the access belonging to priority access group and processes it prior to other accesses.

Holton, D.R.W., Younas, M., and Awan, I.U. developed and evaluated the system that was assumed to be used in e-commerce area[3]. They contended that “the requests received from a Web portal should generally get higher priority as such requests are more likely to lead to purchases”. Received access is stored in the queue corresponding to type of it. If priority of incoming access is higher than that of current processed access then it is suspended and incoming access is processed.

Elnikety, S., Nahum, E., Tracey, J., and Zwaenepoel, W. determined the priority of a process according to the size of job[4]. By lowering the priority of a big job, it raises the overall responsiveness. Cherkasova, L. and Phaal, P. introduced a session based admission control to ensure that longer sessions can be completed[5]. If a server is functioning near its capacity, a new session will be rejected. These researches emphasize to prevent overloading and maintain total throughput at peak rather than to ensure the response time of each classes. Menasèc, D.A., Fonseca, R., Almeida, V.A.F., and Mendes, M.A. present a family of priority-based resource management policies for e-commerce servers[6]. This aims at optimizing revenue/sec instead of focusing on conventional performance metrics. Their target is improvement of responsiveness to clients and single Web server is assumed. In our research, responsiveness is improved by using multiple cache servers, and the quality of resource served by them is raised by priority processing.

3 DESIGN AND IMPLEMENTATION OF PRIORITY ACCESS MECHANISM

Figure 1 shows Priority Access Mechanism we developed. It is implemented based on NAP-Web[7] developed in our laboratory.

Before describing Priority Access Mechanism, we describe NAP-Web. It is the access control mechanism developed as a module of Apache[8]. When response of access is expected to take too long time caused by overload, NAP-Web returns a ticket to notify the next access time to the client which sends the access. When the client re-accesses at the notified time, NAP-Web accepts the access as far as possible.

In order to control accesses, NAP-Web uses Run_Ready (RR), Wait queue (WA) and Re_Access queue (RA) and Next_Wait table. They are shown in left part of Figure 1. RR stores currently processing accesses. Thus the size of RR decides the maximum processing number of accesses. By limiting the size of RR, Web server load can be suppressed. WA stores accesses waiting for processing by FIFO manner. Next_Wait keeps ticket information to identify access whether it is re-accessed or not. RA stores re-accesses by FIFO manner. When RR is not full, the incoming access is entered in RR. When RR is full and all accesses in WA and RA are expected to finish within upper limit time, it is queued in WA. In our experiments the upper limit time is set 4 seconds. Otherwise, the ticket is created, recorded on Next_Wait and returned the client which sends the access. Thus NAP-Web does not pile up accesses waiting for TCP connection, and new connection request can be established immediately. If re-access reaches at the notified time, it is queued in RA. When RR goes to not full, the head of WA goes to RR if

![Figure 1. Priority access mechanism based on NAP-Web](image-url)
WA is not empty, and the head of RA goes to WA if RA is not empty.

We now describe Priority Access Mechanism shown in right part of Figure 1. Accesses are classified into two groups. One is priority access group handled by Priority Access Mechanism. The other is normal access group handled by original NAP-Web. After this we call original NAP-Web “Normal Access Mechanism". Pri_Run_Ready (PRR) stores currently processing priority accesses. Pri_Wait queue (PWA) stores priority accesses waiting for processing. By limiting the number of simultaneous priority accesses under the size of PWA, priority accesses are always processing or waiting with keeping TCP connection. Normal Access Mechanism and Priority Access Mechanism are placed in parallel. The number of current processing accesses is sum of the number of accesses in RR and PRR.

We divert queues of NAP-Web except Next_Wait and RA to Priority Access Mechanism and add selector. By changing the maximum number of access in RR and PRR, system can control computer resources for processing priority access. In our system, source IP address is used to identify priority access. Selector checks source IP address of reached access. If it is one of one of IP addresses of cache servers, the selector passes the access to Priority Access Mechanism. Otherwise, it passes the access to Normal Access Mechanism. In experiments in Section 4, IP addresses of priority client and cache server are hard coded.

4 EVALUATION OF PRIORITY ACCESS MECHANISM

In this section, influence of priority access on response time of normal access, adjustment of the quota of the computer resource to priority access and the effect on response time to clients through cache server are examined.

4.1 Influence of Priority Access on Response Time of Normal Access and Adjustment of the Quota of the Computer Resource to Priority Access

Priority Access Mechanism is examined whether it could adjust the quota of computer resource to process priority access by changing the ratio of the size of RR and PRR. The throughputs of both groups were measured with several combination of the size of RR and PRR.

4.1.1 Experiment Description

Experiment environment is shown in Figure 2. Each client issues the designated simultaneous accesses severally to a bulletin board system (ASKA BBS [9]) on origin server. Each client generates designated number of threads. Each thread accesses repeatedly to origin server simultaneously. The upper limit of the response time of the NAP-Web is set to 4 seconds. The combination of size of RR and PRR used in the experiments is shown in Table 1. (n, m) shows (the size of PRR, the size of RR). The total number of threads on normal clients is 3,000 which is enough number to make origin server stay in overload. The number of threads on priority client is 5 which is enough number to fill PRR. Normal clients start access at the beginning of experiments and priority client starts access at after 200 seconds later. The reason to delay starting time of priority client is that re-access time prediction is disordered just after starting normal access.

Figure 2. Experiment environment for throughput examination

| Table 1. Combination of RR and PRR |
|-----------------|-----|-----|
| RR             | PRR |     |
| 10             | 1   | (1,10) |
| 20             | 2   | (2,20) |
4.1.2 Results of Experiments

Figure 3 shows the response time of normal access. Moving average of response time which window size is 10 seconds is used to plot graphs. Graphs are separated into 2 groups in all parameters. This is because there are 2 kinds of accesses. One succeeds at the first time access, and the other requires re-access. Response time required re-access is sum of the processing time in the first access, the waiting time and the processing time in the re-access. The lower group includes accesses that succeed at the first time, and the upper group includes accesses that succeed by re-access. Response time in lower group is about 4 seconds which is almost equal to upper limit stay in WA and RA. This shows that maximum staying time in WA and RA is not so influenced by priority access. Response time without priority access (0, 20) in upper group is about 20 seconds in all time. However response time after starting priority access (200 seconds point) in upper group lengthens and is disordered. The response time with high percentage of PRR is longer than that of low percentage of PRR.

Priority access throughput (THp), normal access throughput (THn) and total throughput (THt) are shown in Figure 4. They are placed lower, center and upper location, respectively. THt is sum of THp and THn. THt is about 160 in all combinations even if the sizes of the RR and PRR are changed. It shows that THt is not influenced by the size of them. Table 2 shows the percentage of PRR in sum of PRR and RR and average percentage of THp in THt for 600 seconds (between 200 to 800 seconds). We expected that the percentage of THp was roughly equal to the percentage of PRR. The percentage of THp is, however, higher 12.3% to 18.7% than that of PRR. Moreover the percentage of THp with (1, 10) is different from that of THp with (2, 20). On the other hand, the percentage of THp with (2, 20) is higher than that of THp with (1, 20), and the percentage of THp with (2, 10) is also higher than that of THp with (1, 10). Magnitude relation between percentage of THp and percentage of PRR with same size of RR is preserved. This result shows that it is possible to adjust the quota of computer resource to process priority access but it is insufficient. Improvement of adjustment precision is future work.

4.2 The Effect on Response Time to Clients Through Cache Server

The purpose of this experiment is to confirm that Priority Access Mechanism can reduce response time to clients through cache server. As shown in Figure 5, we use origin server with our mechanism, cache server, direct clients accessing origin server, and indirect client accessing cache server. While origin server is overloaded by numerical simultaneous accesses from direct clients, indirect client accesses to cache server. The expiration date of cache is set a few seconds for caching dynamic contents. The response time to direct and indirect clients is measured.
4.2.1 Experiment Description

Cache server is configured as a reverse proxy of origin server. To add the reverse proxy function and the caching function to the cache server. We use “mod_proxy” and “mod_disk_cache” modules of Apache, respectively. All clients access ASKA BBS with read only mode to serve same contents to both clients. The expiration date of cache is set to 5 seconds. We think it is short enough to cache BBS type contents. The total number of threads on direct clients is 2,000 which is enough number to make origin server stay in overload. The number of threads on indirect client is 300 with which cache server does not stay in overload.

4.2.2 Experimental Result

Figure 6 shows response time on indirect client. Graphs are separated into 2 groups because of existing two kinds of response. The lower group includes response time to return cached content directly. The upper group includes response time with cache update. The average cache update time is about 0.25 seconds. We think that this time enables update at several seconds interval and dynamic read only contents can be cached.

Figure 7 shows response time on direct client. Graphs are separated into 2 groups and response time of both groups is longer than 4 seconds like Figure 3. This is because that direct access is controlled by Normal Access Mechanism. Response time to direct access is too longer than that of indirect access.

Next, we experiment using origin server without Priority Access Mechanism. Original NAP-Web should be used but cache server does not support re-access. Thus we use original Apache for the experiment. Figure 8 shows response time on indirect client. Similar to Figure 6, response time is separated into 2 groups, but response time with cache update is about 7 seconds which is about 25 times longer than response time with cache update using Priority Access Mechanism.

Figure 9 shows response time on direct clients.
Range of response time is about 5 to 10 seconds. This range is same as range of response time on indirect client with cache update. It is thinkable that the response time to indirect client with update is similar to graphs in Figure 7 when original NAP-Web used.

These results show the effectiveness Priority Access Mechanism.

5 DEVELOPMENT OF PRIORITY ACCESS MANAGEMENT MECHANISM

As mentioned at the end of Section 3, experiments in Section 4 are achieved using hard coded IP addresses to recognize priority accesses. In order to append and remove cache servers provided on cloud environment dynamically, we develop Priority Access Management Mechanism which registers appended cache server to Priority Access Mechanism as priority host and deletes registration information of remove cache server. By registering clients that owner are administrator and user who want to update his page using Web interface, they can get fast response.

In the following subsections, target recognition method, design and function evaluation of Dynamic Registration Method are described.

5.1 Consideration of Target Recognition Method

IP address and cookie can be used to recognize whether an access is priority access or not. In our implementation of Priority Access Mechanism used in Section 4, IP addresses of priority client and cache server are stored in IP address array (IP-Array) statically, and source IP address of arrival access is checked whether it is included in IP-Array or not. It is easy to implement the method using IP address by appending a function which registers IP address of appended priority host with IP-Array and deletes IP address of removed priority host from IP-Array.

If a client that does not have global IP address and access Web server using NAT, and registered as a priority client, accesses from other clients that share global IP address with registered clients are also recognized as priority access. In such case, the method using IP address cannot be used. This problem can be solved by using cookie. In order to handle cookie, we introduce priority key array (PC-Array) and scan function of PC-Array into Priority Access Mechanism. However, this method requires that each host needs to correspond to cookie. Therefore, it may not be able to be used depending on the cache server.

Since cache servers always have global IP address in our distributed Web system, we decide that the method using IP address is used for cache server and the method using cookie is used for clients. Registration and delete of IP address of cache server are done by cache server management system. Registration and delete of cookie are done by users having their page on Web server.

5.2 Design

Priority Access Management Mechanism has an Authentication CGI, an API for registering and deleting priority IP address and cookie, and a status page for showing the registration list. Status page is used for managing.

Authentication CGI is the interface of Priority Access Management Mechanism, and used for registers cookie. It examines whether user who want to register his host is legal or not, and then requests priority key to API and returns priority cookie including priority key as shown in Figure 10. Priority key has an expiration date, and it is deleted automatically if there is no access for a certain period of time.

API has interfaces that generate, register and return priority key, and register and delete
5.3 Function Evaluation

Functions of the implemented mechanism are examined. We create status page which shows the number of accesses in each queue. By accessing the status page as the destination page, we can confirm the access is processed by which mechanism; Normal Assess Mechanism or Priority Access Mechanism.

Figure 11 shows the result of cookie version. No client is registered at staring experiment. Figure 11-(a) shows the status page accessed before registration. Strings at left side denote queue name and figures at right side denote the number of accesses included in corresponding queue. The access is processed by Priority Access Mechanism because RunReady includes one access and pre_RunReady includes no access. Figure 11-(b) shows the status page accessed after registration. the access is processed by Normal Access Mechanism because RunReady includes no access and pre_RunReady includes one access. These results show that a client can get priority cookie from Priority Access Management Mechanism and access from the client is processed by Priority Access Mechanism.

We also experiment using IP address version. The same results as the experiment using cookie version is obtained before and after registration of IP address.

There results indicate that Priority Access Management Mechanism works fine.

(a) Access before registration (b) Access after registration

Figure 11. Queue information

6 CONCLUSION

We have developed Priority Access Mechanism for access from cache server and Priority Access Management Mechanism for registering priority hosts dynamically. We confirm that cache can be...
renewed in a short time by Priority Access Mechanism and then short term cache for dynamic contents could be possible. And the mechanism can adjust the quota of the computer resource to handle access from cache server. However adjustment precision is insufficient. We also confirm that Priority Access Management Mechanism can register priority hosts dynamically and accesses issued from registered hosts are processed by Priority Access Mechanism.

The feature work is as follows.

- Improvement of adjustment precision.
- Evaluation using more realistic situation, e.g., fluctuating the number of accesses over time.
- Evaluation of effectiveness of caching dynamic contents.
- Implementation of deleting function of Priority Access Management Mechanism.
- Integration of Priority Access Mechanism with our distributed Web system.

REFERENCES