L-043

Simulation of Adaptive Network against Flash Crowds

Chenyu Pan† Merdan Atajanov† Toshihiko Shimokawa‡ Norihiko Yoshida†

1. Introduction

With the rapid spread of information and ubiquitous access of browsers, flash crowds, triggered by a sudden, unanticipated surge in user request rates, overwhelming the target web site temporarily unavailable, have become the bane of many Internet websites.

In this paper, we focus on modeling and simulating a self-tuning adaptive network, FCAN, which dynamically transits system structure between peer-to-peer(P2P) and client-server(C/S) configurations to reach the optimum efficiency in protecting website from flash crowds. FCAN has been previously introduced elsewhere [1, 2]. As an updated and improved version, this paper modifies the basic design of FCAN by adding volunteer clients as peer members to assist P2P functions and introduces the conception of implicit P2P overlay. Moreover, the paper presents the preliminary simulation results to confirm the design.

The rest of the paper follows as: Section 2 analyzes the motivation of the research. Section 3 presents the improved design. Simulation results are analyzed in Section 4. Discussions are described in Section 5 and the last is the conclusion.

2. Motivation

There are several approaches to handle the flash crowds. A straightforward but costly approach is to provision the accessibility based on peak demand or increase the server locations by CDN service. However, since the duration of flash crowds is relatively short, these server-side solutions result the provisioned resource idle most of the time and are not justified to small websites.

An alternative is to have the clients form a P2P overlay to share the hot objects among themselves. The limitations of this schema is losing client transparency and remaining low efficiency when facing the leaving of flash crowds [3].

Our idea of FCAN is to adapt the network structure between P2P and C/S configurations. FCAN employs an Internet-wide infrastructure of cache proxy servers to perform P2P functions in dealing with the flash crowds effects and gets it out of the way when normal C/S architecture works well. Through this way the unjustified overprovision can be avoided and the overheads caused by P2P functions can be minimized. Besides, building P2P overlay on the cache proxy layer also remains the client transparency.

3. Design Description

Given the short duration of flash crowds, FCAN remains C/S architecture at most of the time and enables cache proxies to perform P2P search only during flash crowds. To some extent, FCAN organizes a group of forward cache proxies into a temporary and wide-area based layer of reverse cache proxy.

Figure 1 shows the changing architecture of FCAN. Different from our previous design [1], to better provide the service, we also welcome the participations from volunteer clients as peer members. In the figure, Member Server, Member Peers, including Member Cache Proxy (CP) and volunteer client, and special DNS are shown in deep color to distinguish from other clients who connect to the web server directly or through common cache proxy.

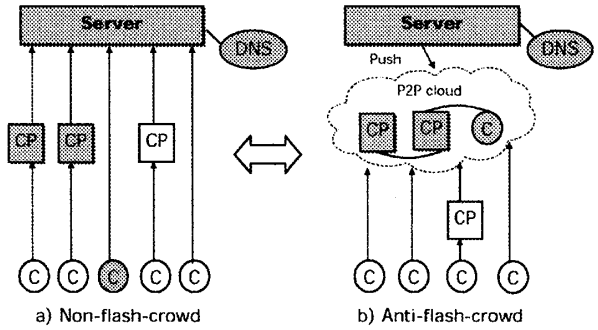

Figure 1: Changing architecture

In peaceful time, Member Server and Member Peers do little more than what normal ones do. Once a flash crowd comes, Member Server detects the increase in traffic load. It modifies its home DNS entries to the address of P2P cloud, here the address of P2P cloud is a list of the address of all available Member CP. Then Member Server triggers all Member Peers to form the P2P cloud through which all requests are conducted instead of bugging the origin server. And the modified DNS entries are propagated through Internet to make subsequent client requests redirected from original server.

In addition, research[4] shows over 60% of objects are new to a flash crowd, which may not been cached at the beginning stage. Member Server pushes these objects to the cloud by connecting to just one or two Member Peers. The pushed objects will soon be propagated among the cloud because of the P2P functions and the high demands.

For detail design issues, please refer our previous papers [1, 2].

4. Implementation Overview

We put special wrappers to intercept the requests on normal web servers and cache proxies, and employ special DNS server to form FCAN network system.

The implementation of wrappers mainly includes P2P overlay construction, load observation, and adaptive triggering. P2P overlay construction should be fast started and stopped with the change of the network conditions. Different from our previous researches, we introduce the implicit P2P overlay which can be either applied with first generation, such as PROOFS[3], or second generation P2P system, such as Tapestry[5]. The relationship among peers are built through normal communication processes and are not removed even after the transition. Member peers exchanges the existence through Member Server and maintains an implicit P2P overlay continuously. The overlay is invoked

†Saitama University
‡Kyushu Sangyo University

to be explicit only during flash crowds and the P2P search is enabled as well. Load observation measures the system condition and provides the adaptive triggering with the observed data. The load observation on peer reports the real time traffic load of itself to Member Servers periodically through IP address based communication, while the load observation on server collects the traffic information of itself as well as the whole P2P cloud and makes a global decision for adaptive triggering. Adaptive triggering is a server-triggered adaptation.

All the wrappers will be designed as a module for the popular Apache web server and Squid cache proxy. Both Apache and Squid are built from a modular design, which is easy to add new features. Thus converting normal elements with Member features should require no more than compiling a new module into Apache or Squid and editing a configuration file.

To implement special DNS server, we adopt TENBIN system [6]. TENBIN is our research product, and already used in practice, for example, "Ring Servers" and "LIVE! ECLIPSE" projects. With TENBIN, we can dynamically modify DNS lookup entries and configure special policy to achieve load balance among P2P cloud. The details of TENBIN have already been presented elsewhere [6].

5. Preliminary Simulation

In this section, we present the preliminary simulation results. The simulator is a home-grown, thread-based, java program and mainly composed of five parts:

1. A package of core classes including the main roles of system such as Member Server.
2. A flash crowds traffic generator which models a flash crowd with shock level and three phases, ramp-up phase, sustained load phase and ramp-down phase.
3. Underlying network architecture which supports high-level application with the TCP and UDP communication and creates unique network address for each node.
4. Network monitors which collects the load information from active roles periodically and writes the load data to the log files.
5. Console User Interface which enables parameters configuration and runs test case for simulator user.

5.1 Configure Parameters

Table 1 shows the detail parameters configuration. We configure the network composed of ten Member CP and one Member Server. Clients are created and disappeared during the running of simulation.

There is no hot object copy on any Member CP at the beginning. The P2P search algorithm remains a TTL scoped one and the neighbor-ship isn't removed after a flash crowd is over so that the implicit P2P overlay continues.

Given a normal request rate 8 requests per second, the threshold is set to 20 requests per second. However, the server may still suffer from increased high request rate for several seconds since the transition takes some time to complete. And the restore C/S threshold is set to the value of 80% of the P2P threshold.

We configure the shock level of flash crowd to 10, thus the peak load reaches to 10 * 8 requests/second.

Table 1: Simulation Parameters

Description	Value
Number of peers	10
Number of peers receiving push object	1
P2P search ttl	2
P2P search fanout	1
Number of flash objects	1
Number of common objects	9
Shock Level of flash crowd	10
Normal traffic rate (req/sec)	8
Threshold for alarm (req/sec)	20
Threshold for restore (req/sec)	0.8*20
Starting time of flash crowd (sec)	10

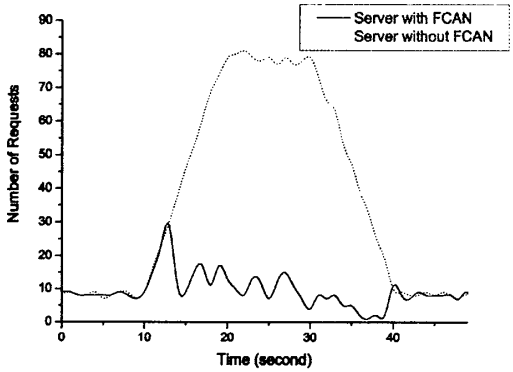

Figure 2: Comparison of load on servers

We start the flash crowd at the 10th second and the ramp-up duration l1 , sustained load duration l2, ramp-down duration l3 are computed by following formulation.

l1 = Math.log(10)/Math.log(1+shocklevel) * 10;
l2 = Math.log(1+shocklevel)/Math.log(10) * 10;
l3 = Math.log(1+shocklevel)/Math.log(10) * 10;

5.2 Simulation results

Figure 2 illustrates the comparison of two servers which suffered from modeled flash crowd traffic simulated by traffic generator, where the traffic duration was 50s, flash crowd started at the 10th second, the ramp-up phase lasted for 9.6s, starting from 8 req/sec to 80 req/sec, the 80 req/sec sustained load phase lasted for 10.4s and the ramp-down phase lasted for 10.4s. Compared with the server without FCAN, the server with FCAN controls the request rate under 30 requests/second and the load degraded during the flash crowd period because the P2P cloud cached objects for the clients. Therefore, the adaptive transition does alleviate the flash traffic from server side.

The two transition points are around the time of 14s and the time of 40s, as Figure 3 shows. At the time of 12s the number of requests on Member Server exceeded the threshold 20 and reached 23. It took 2 seconds to complete the transition. At the time of 14s P2P cloud began to receive the redirected client requests. The time for restoring the system from P2P to C/S configuration is relatively short. At the time of 39s the average load on P2P cloud drop to 1.5, below the threshold 1.6. One second passed, it restored the system and no

Table 2: Detail data at transition points

Time	MS	CP1	CP2	CP3	CP4	CP5	CP6	CP7	CP8	CP9	CP0
12	23	0	0	0	0	0	0	0	0	0	0
13	29	0	0	0	0	0	0	0	0	0	0
14	11	3	1	4	9	1	3	1	2	3	0
39	2	3	1	1	0	2	2	1	2	2	1
40	11	0	0	0	0	0	0	0	0	0	0

Table 3: Comparison of Handled Requests by each individual

	FC	MS	CP1	CP2	CP3	CP4	CP5	CP6	CP7	CP8	CP9	CP0
Total requests	1957	473	140	128	166	176	156	130	140	137	167	144
Peak request rate	80	29	13	10	11	14	15	11	15	11	11	14
Ave. request rate (50s)	39.14	9.46	2.80	2.56	3.32	3.52	3.12	2.60	2.80	2.74	3.34	2.88

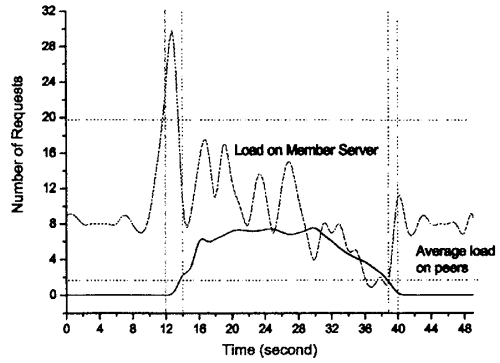

Figure 3: Average load on peers

subsequent request was redirected to the cloud. Table 2 lists the detail request load information of each individual at those two transition points.

Table 3 gives the handled request information of Member Peers as well as Member Server. The FC column is the modeled flash crowd traffic that impact on the normal server without FCAN. From these data, we can draw the conclusions that: 1) the load distributed among P2P cloud is balanced by each peer, 2) each member role is under an acceptable average load and 3) server protected by FCAN is free from the sustained high flash traffic.

6. Discussions

6.1 Mixed-Mode Operations

In reality, each cache proxy serves for several content servers, and there is a case that any server suffers from flash crowds while the others do not. Therefore, each Member CP has the functionality of mixed-mode operations for the normal C/S mode and the anti-flash-crowd P2P mode. The modes are switched according to requested contents.

6.2 Peer Load Control

Each Member Peer spares a part of its CPU cycles and disk space to deal with the redirected requests. We should limit the load on any participating peer so as not to overload it as a secondary victim of a flash crowd. Thus, the capacity of P2P cloud absorbing flash traffic is directly proportional to the number of Member Peers which participate into the task of load distribution.

7. Conclusion

This paper updates the design of Flash Crowds Adaptive Network, discusses the implementation issues and presents the preliminary simulation results. The results validate the system's correctness. The next steps in this research involve the closer study of possible solution to non-cacheable objects, detailed component designs for wrappers, and a real network based evaluations.

References

[1] C. Pan, M. Atajanov, T. Shimokawa and N. Yoshida. Design of Adaptive Network against Flash Crowds. FIT2004. September 2004.

[2] C. Pan, M. Atajanov, T. Shimokawa and N. Yoshida. Flash Crowds Alleviation via Dynamic Adaptive Network. IC2004. October 2004.

[3] A. Stavrou, D. Rubenstein and S.Sahu. A lightweight, robust P2P system to handle flash crowds. IEEE Journal on Selected Areas in Communications, Vol.22, No.1,January 2004.

[4] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In Proceedings of the 11th International World Wide Web Conference, pp. 252–262. IEEE, May 2002.

[5] B. Y. Zhao, J. Kubatowicz, and A. D. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. U. C. Berkeley Technical Report UCB/CSD-01-1141, April, 2001.

[6] T. Shimokawa, Y. Koba, I. Nakagawa, B. Yamamoto and N. Yoshida. Server Selection Mechanism using DNS and Routing Information in Widely Distributed Environment (in Japanese).Trans. IEICE, Vol.J86-B, no.8, pp.1454–1462 (2003)