A Request Distribution Algorithm for Web Server Cluster

Wei Zhang
School of Computer Science and Engineering, Beihang University, Beijing, 100191, China
State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing 100044, China
zhangwqh@cse.buaa.edu.cn

Huan Wang, Binbin Yu, Wei Xu, Mingfa Zhu, Limin Xiao, Li Ruan
School of Computer Science and Engineering, Beihang University, Beijing, 100191, China
{zhumf, xiaolm,ruanli}@buaa.edu.cn

Abstract—**With the explosively increasing of web-based applications' workloads, Web server cluster encounters challenge in response time for requests. Request distribution among servers in web server cluster is the key to address such challenge, especially under heavy workloads. In this paper, we propose a new request distribution algorithm named llac (least load active cache) for load balancing switch in web server cluster. The goal of llac is to improve the cache hit rate and reduce response time. Packets are parsed in IP level, and back-end servers are notified to cache hot files using link change technology, neither changing URL information nor modifying the service program. This avoids switching overhead between user mode and kernel mode. The load balancing switch directly creates connection with the selected server, avoiding migrating connection overhead. This policy estimates the current composited load of each server and selects the server with the least load to serve the request. It also improves the resource utilization of web servers. Experimental results show that llac achieves better performance for web applications than wrr (weight round robin) which is a popular request distribution.**

Index Terms—**Web Cluster, Request Distribution, LLAC**

I. Introduction

The enormous growth of the internet industry introduces web-based application as popular demanding programs. Users are becoming increasingly reliant on the web for their daily activities such as electronic commerce, on-line banking, stock trading, reservations and product merchandising. Therefore the performance of a web server system plays an important role in success of many internet related companies. Traditionally, a single server machine can only handle a limited amount of requests and can't scale up with demand. The better way to cope with growing processing demands for web servers is by adding more hardware resources instead of completely replacing one server with a faster one [1]. More and more web sites use a web cluster, composed of a front-end request dispatching server, also called load balancing switch, and several back-end servers handing requests. By distributing requests from clients to separate servers for load balancing or load sharing, web cluster have proved to be a better solution than using an overloaded single server. Due to various technical issues regarding the management of a web server cluster, request distribution algorithms (which are implemented in the load balancing switch) are particularly important to boost the performance of cluster web servers [2]. The ratio of the peak load to light load for internet applications is usually on the order of 300% [19]. J.C.Mongul said [3], web site happened to collapse mostly because of popular and hot event access. A famous example, the normally well-provisioned Amazon.com site suffered a forty-minute down time due to an overload during the popular holiday season in November 2000. Popular web sites often face the challenge to deal with huge amount of requests in short time. This paper addresses the problem of request distribution so that web server cluster can serve its peak workload demand. We simultaneously use client-side and server-side information to select server, and avoid switching overhead between user mode and server mode. We also avoid migrating connection overhead. We present a new request distribution algorithm with several contributions in which as follows: Firstly, design combined load model based on collection of typical load information. This model is used with online measurements of load information to estimate the processing capacity of web servers. This gives us reliable load descriptors for web servers which are used in the decision making process of the request distribution algorithm. Secondly, in order to increase the speed of accessing the popular or hot files, our approach resorts to active caching technology. Packets are captured and analyzed using netfilter mechanism in IP level, avoiding switching overhead between user mode and server mode and migrating connection overhead. The active caching technology does not modify URL or server program, resorting to link change technology to put hot files in memory file system. Finally, we propose and implement a novel request distribution algorithm which works on the basis of the composited load and file access frequency. We call this novel request distribution algorithm for llac(least load and active caching) shortly.

The rest of the paper is organized as follows. Section II discusses the related work. In Section III we propose request distribution architecture and algorithm, and then we discuss each module separately. Section IV describes the experimental results of a web cluster prototype using llac. Section V concludes the paper.

II. Related works

Numerous dispatching algorithms are proposed for web server cluster. We can classify dispatching algorithms as layer-4 and layer-7 algorithms.

A layer-4 algorithm only considers web server-side information, but doesn't use client-side information. In this approach, clients directly create connection with the selected server. This algorithm is easy to implement, but cannot make good use of server's resources according to the customer's request. It includes random policy [4], round-robin policy [4], weight round-robin policy [5], least connection policy [6], fast response time [6] and so on. Random and round-robin are easy to implement, but they don't consider servers capacity. This can easily lead to unbalance. Wrr associates an evaluated weight with each server node in a cluster which is proportional to the server's capacity. Initial weight is set by the administrator, disturbing by human factors. Least connection doesn't consider that each request may have different response time and different demand for resources. Fast Response Time is influenced by the network environment, so can't evaluate the performance of a web server effectively.

A layer-7 algorithm not only considers server information, but also can use client's user level information, such as session identifiers, type of URL, cookies and so on. However, clients need to create TCP connection with the load balancing switch in order to analyze information of customer. This involves to two copies of packets between user space and kernel space. As customers firstly establish connections with the load balancing switch, so the connections need to be migrated to the selected server. Migrating connections are very time-consuming and consume large amounts of system resources. Layer-7 algorithms can consider more information deciding to select which server to response to a request and make good use of server resources, in particular cache resource. However, they require migration of connection and copy of packets between user space and kernel space, bringing a certain degree loss of performance. Examples of the layer-7 include LARD (locality-aware request distribution) [7], WARD (workload aware request distribution) [8], CAP (client aware policy) [9] and so on. LARD is well known dispatching policy that aims to improve cache hit rate in web server. In LARD policy, the load balancing policy dispatches the request of the same web object to the same back-end web server. However, LARD may lead to load unbalancing due to different popularity of web pages. WARD is static partitioning that assigns dedicated servers to specific groups of requests. Although this policy is useful from the system management point of view and achieves a higher cache hit rate, it does have poor server utilization. Degradation in the utilization is due to resources that are not utilized and cannot be shared among all of the clients. The main goal of CAP is to improve load sharing in web clusters that provides multiple types of services. The load balancing switch classifies requests from clients into four classes: normal, CPU bound, disk bound, and CPU and disk bound. However, requests with the same type might consume different amounts of resources.

Considering the shortcomings of the above methods, we propose llac. Using netfiler mechanism to intercept packets and analyze the URL information in IP level, Load balancing switch notifies the back-end server cache hot files, selects least-load server to process requests. We use both client-side and server-side information, avoid switching user mode and kernel mode and migrating TCP connection overhead. In case of hot files, use cache to improve response time.

III. LLAC for web cluster

In the sites, most of the crashes occur during the hot visit. Therefore, increase accessing speed of hot files eases the pressure on the sites to a large extent. The main objective in designing such an algorithm is to minimize the average response time of popular or hot requests (proactive cache hot files, read them from memory file system, so reducing disk write times), make good use of server resources in the cluster and increase the utilization and throughput of cluster web servers. In this regard, we use a linear model to compute each server's composite load, which helps to decide which should be chosen to serve request. The module of llac uses information of clients and servers to make request distribution. The load balancing switch parses packets in IP layer using netfilter mechanism, records packet access frequency and informs web servers to cache popular or hot files. Fig. 1 shows system components we design.

A. Load Collection

The Load Collection is responsible for tracking the processor, network and memory usage of web server. We gather resource utilization traces by running a set of microbechmarks. The full list of metrics is shown in TABLE I. These statistics can all be gathered easily in Linux with the sysstat monitoring package [10]. We focus on this set of resource measurements since they can easily be gathered with low overhead and are representative of estimating the performance of the web server [11]. Since these traces must also be gathered from the live application, it is crucial that a lightweight monitoring system can be used to gather data. To improve performance, we create a thread for every load indicator in parallel to execute. The load collection tracks the usage of each resource over a measurement interval and reports these statistics to the load calculation at the end of each interval.

B. Load Calculation

This section describes how to create models which characterize the relationship between a set of resource utilization metrics gathered from an application running on the web server and the composited load. The model creation employs a component which is a linear equation.

Using values from the load collection module, we form an equation which calculates the total load as a linear combination of the different metrics.

$$T = \alpha_0 + \alpha_1 * U_1 + \alpha_1 * U_2 + \cdots + \alpha_9 * U_9 \quad (1)$$

Where

- U_i is a value of metric collected for a benchmark executed in the web server;
- The set of coefficients $\alpha_0, \alpha_1, \cdots, \alpha_n$ is the model that describes the relationship between the total load and Resource Utilization Metrics. Unfortunately, finding a set of good parameters is a rather empirical job, with very little support from theory. The main objective is to tune the parameters to achieve good system performance, without asking too many questions about why it works well. Often, it is just a matter of “let’s try this approach and see what happens”.
- This is the total load of the web server.

Load calculation module passes the results to load management module located in the load balancing switch. Web Server adopts active push method to report their composited load. Pushing way than active asking can further reduce the burden of the load balancing switch. Also, Using UDP unicast data transmission can reduce the burden on the network bandwidth.

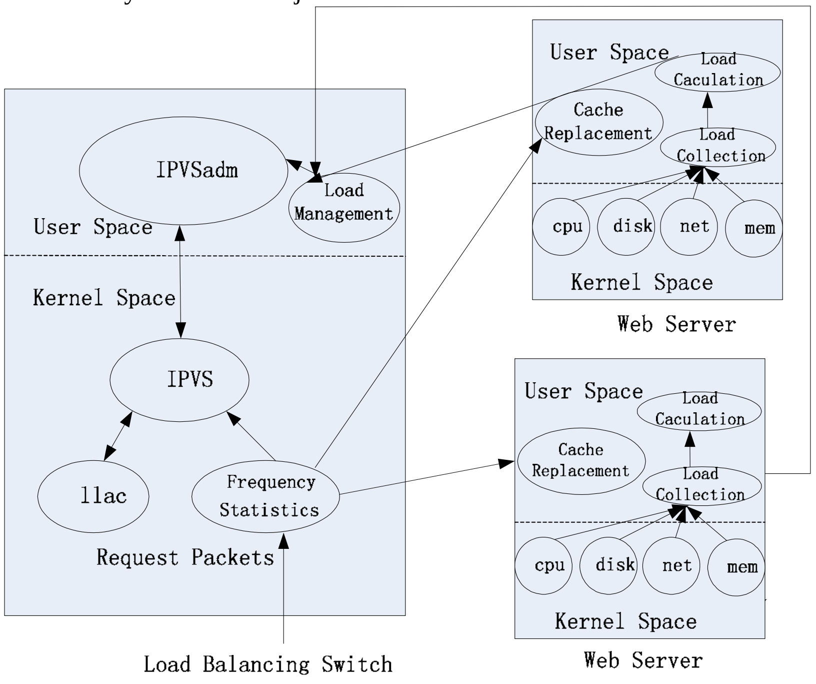


Figure1. System Architecture

TABLE I. RESOURCE UTILIZATION METRICS

CPU	Memory	Network	Disk
User Space %	Memory Used %	Rx packets/sec	Read req/sec
Kernel Space %	Swapper Used %	Tx packets/sec Rx bytes/sec	Write req/sec Read blocks/sec
IO Wait %		Tx bytes/sec	Write blocks/sec

C. Load Management

In the listening state, if it receives the server’s composited load information, it creates a child thread and notifies the current load value to llac module located in the kernel space using IPVSadm management tool.

Frequency statistics module is mounted on the IP_LOCAL_IN in the Netfilter[12] in order to parsing the request packets and count the access frequency statistics. The priority of the frequency statistics function must be higher than the IPVS, otherwise the request will be forwarded out at this point, leading to not reaching frequency statistics function. Meanwhile, we use hash lists in order to raise the speed of accessing and searching. They link together through the general list head pointer inside the structure. We sort the file from more to less according to access frequency, through sorted_list pointing to the sorted list (Fig. 2) so that we can easily get hot file information.

D. Cache Replacement

We use memory file system divided from memory space for hot file cache. Our novelty lies in using link change technology to modify file location on disk (changed to symbolic link) pointing to the location on memory file system caching the file. It brings many benefits. For example, we do not need to modify the URL information. Also, service program does not require modification, we can access the cached file from the memory space. Whether the file is on disk or in memory file is transparent to the user or service procedures.

Due to the size limit of RAM resource, when memory space in memory file is not sufficient to accommodate the needs of caching file, cached files need to be replaced out from the cache using replacement policy. We use the following cache replacement strategy. We sort the files according to the ratio of file access frequency and file size. When need to be replaced, give priority to small ratio. Follow this way, the access frequency which is low and the size which is large will be replaced out. To a certain extent, this improves the cache hit rate, also improves the cache utilization.

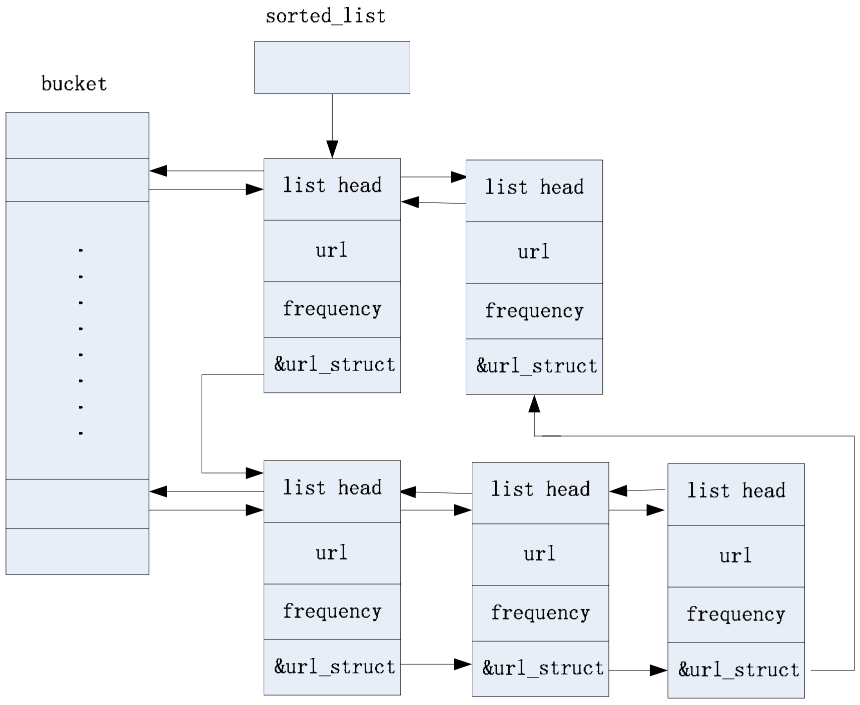


Figure2. File Frequency Statistics

IV. EXPERIMENTAL RESULTS

To analyze the proposed dispatching algorithm, it is implemented on a web server cluster. We implement the experimental testbed with hardware and software configurations as described below.

A. Hardware setup

The web server cluster consists of a load balancing switch node, connected to the web server nodes. All the nodes are connected through a high speed gigabit LAN switch. The distributed architecture of the cluster is hidden from the clients via a unique virtual IP address. The hardware environment is shown in TABLE II.

TABLE II. HARDWARE ENVIRONMENT

	CPU	Memory(GB)	HD	NIC
Front-end	Intel(R) E5345 2.33GHz 8cores	DDR2 4	60GB	80003ES2LAN
Back-ends (1-4)	Intel(R) E5345 2.33GHz 8cores	DDR2 4	60GB	80003ES2LAN

B. Software Setup

TABLE III. SOFTWARE ENVIRONMENT

	OS	Kernel	IPVS	Web server	Benchmarks
LB switch	Red Hat Linux5.0	2.6.18	1.0.4	————	————
Web server	Red Hat Linux5.0	2.6.18	——	Apache 2.0.40	————
Client	Red Hat Linux5.0	2.6.18	——	————	WebBench 5.0

TABLE III shows the experimental software environment. All the machines in the cluster run Linux kernel 2.6.18 as their operating system, and the load balancing switch uses IPVS for request dispatching. We use Apache 2.0.40 for HTTP service installed as the web server. HTTP/1.1 connection is applied. In addition, all clients have WebBench[13] installed for request proposing.

WebBench is a performance testing software for web servers, including both the controller and clients. The controller is able to control clients for proposing requests, to record and summarize the experimental data, and then output the experimental results. In addition, WebBench can control the mixed ratio of request types transmitted from clients by the programmable workload.

We perform all experiments to analyze the system performance under different ratios of request types (e.g. different localities of hot Web pages). We also create a workload generator to generate a synthetic workload for various ratios of request types. The performance metrics we used are the requests per second (req/s) megabits per second (Mbps) and number of successful requests, which are the experimental results summarized and reported by WebBench.

C. *Experimental evaluation*

In this section, we present performance evaluation of our proposed llac request distribution algorithm. In this test, WebBench is used and hot Web pages are built from the requested web pages of the default workload. Furthermore, we prepare the click through rate (CTR) with 20%, 40%, 60% and 80%, change the percentage of the hot web pages in requested web pages to 10%, 20%, 30% and 40%. We compare the experimental results with that of wrr. We also compare llac with only using ll or ac. Fig. 3, 4, and 5 shows that our llac outperforms wll, ll, and ac.

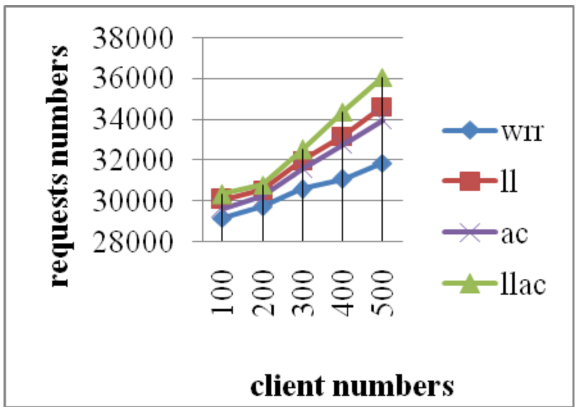


Figure5. number of successful requests

The reason that llac policy performs better than wrr, ll is because the llac policy uses frequency-based mechanism to achieve high cache rates of servers. The reason that llac policy performs better than ac is because the llac policy considers server's current load, assessing the current load and selecting the appropriate node to response the request.

Experimental results demonstrate that when the web server cluster is under heavy load, the llac policy can handle more requests and show better performance.

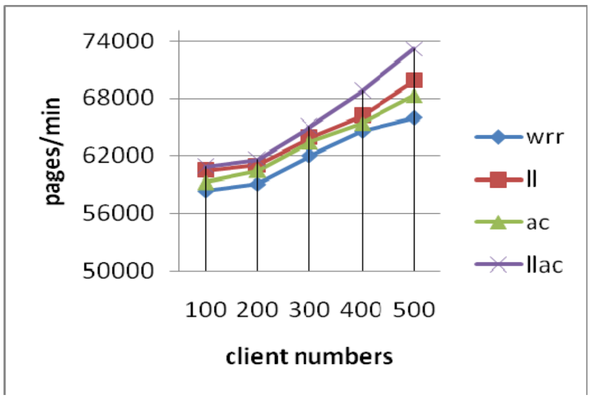


Figure 3 number of requests per minute

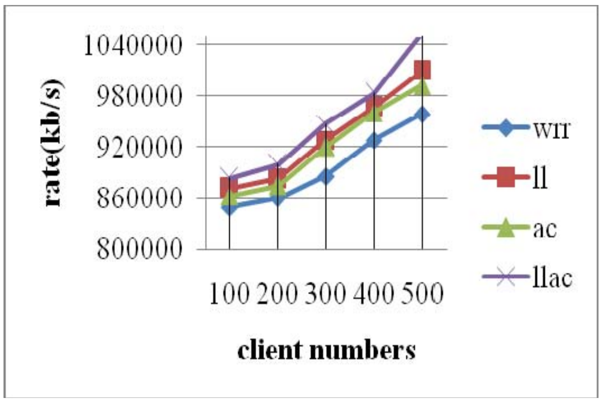


Figure4. data transfer per second

V. SUMMARY

This paper presents a new request distribution algorithm for web server cluster, called llac. This research focuses on reducing hot files access time and uses the resources of web servers more efficiently. Our experimental results show that our proposed llac policy can get better performance than wrr, ll, and ac under heavy load condition. This policy reduces response time especially for hot files, because hot files are retrieved from memory. The node with least load is selected to serve the request so that it results in resource utilization getting better used. In future work we plan to experiment with more benchmark to further verify effectiveness of llac.

ACKNOWLEDGMENT

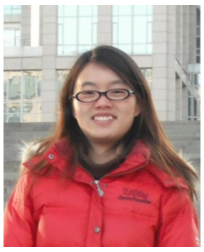
This study is sponsored by the fund of the State Key Laboratory of Software Development Environment under Grant No. SKLSDE-2009ZX-01, the National Natural Science Foundation of China under Grant No. 61003015, the Doctoral Fund of Ministry of Education of China under Grant No. 20101102110018 and the Fundamental Research for the Central Universities under Grant No. YWF-10-02-058.

REFERENCES

[1] A. Chandra, P. Pradhan, R. Tewari, S. Sahu, P. Shenoy. "An observation-based approach towards self-managing web servers", Computer Communications, 2006, pp1174-1188.
[2] V. Cardellini, E. Casalicchio, M. Colajanni, S. Tucci, "Mechanisms for quality of service in web clusters", Computer Networks, vol.37, No.6, 2001, pp761-771.
[3] M.E. Crovella, A. Bestavros. "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", IEEE/ACM Transactions on Networking, vol.5, No.6, 1997, pp835-846.
[4] V. Cardellini, E. Casalicchio, M. Colajanni, and P.S. Yu. "The State of the Art in Locally Distributed Web-Server Systems", ACM Computing Surveys, vol.34, No.2, 2002, pp 263-311.
[5] M. Andreolini, E. Casalicchio. "A cluster-based web system providing differentiated and guaranteed services", Cluster Computing , vol.7, No.1, 2004, pp7-19.
[6] E. Choi. "Performance test and analysis for an adaptive load balancing mechanism on distributed server cluster systems", Future Generation Computer Systems, No.20, 2004, pp 237-247.
[7] V.S. Pail, M. Aront, G. Bangat. "Locality-Aware Request Distribution in Cluster-based Network Servers", ACM SIGOPS Operating Systems Review, USA:ACM , 1998, pp205-216.
[8] L. Cherkasova, M. Karlsson. "Scalable Web Server Cluster Design with Workload-Aware Request Distribution Strategy WARD", Advanced Issues of E-Commerce and Web-Based Information Systems, Washington:IEEE Computer Society, 2001, pp212-221.
[9] E. Casalicchio, M. Colajanni. "A client-aware dispatching algorithm for Web clusters providing multiple services", The International World Wide Web Conference Committee (IW3C2), 2001, pp535-544.
[10] Sysstat-7.0.4. http://perso.orange.fr/sebastien.godard/
[11] M. Andreolini , S. Casolari , Michele Colajanni. "Models and framework for supporting runtime decisions in Web-based systems", ACM Transactions on the Web (TWEB), vol.2, No.3, 2008, pp1-43.
[12] CHRISTIAN BENVENUTI. Understanding LINUX NETWORK INTERNALS. 2006.
[13] http://linux.softpedia.com/get/System/Benchmarks/Web-bench-1378.shtml
[14] M.L. Chiang, Y.C. Lin, L.F. Guo. "Design and implementation of an efficient web cluster with content-based request distribution and file caching", Journal of Systems and Software, vol.81, No.11, 2008, pp 2044-2058
[15] S. Sharifian, S.A. Motamedi, M.K. Akbari. "A content-based load balancing algorithm with admission control for cluster web servers", Future Generation Computer Systems , vol.24, No.8, 2008, pp775-787.
[16] M.L. Chiang, C.H. Wu, Y.J. Liao, Y.F. Chen. "New Content-aware Request Distribution Policies in Web Clusters Providing Multiple Services", Proceedings of the 2009 ACM symposium on Applied Computing, USA:ACM, 2009, pp79-83.
[17] Z.Y. Xu, J.Z. Han, L. Bhuyan. "Scalable and Decentralized Content-Aware Dispatching in Web Clusters", IEEE International Performance, Computing, and Communications, Washington:IEEE Computer Society, 2007, pp202-209.
[18] Y.K. Chang. "Fully Pre-Splicing TCP for Web Switches", Proceedings of the First International Conference on Innovative Computing, Information and Control, Washington:IEEE Computer Society , 2006, pp737-740.
[19] S. Chase , D.C. Anderson. "Managing energy and server resources in hosting centers", In Proc. of the eighteenth ACM symposium on Operating systems principles, 2001, pp103-116.
[20] Tarek F. Abdelzaher, Kang G. Shin, and Nina Bhatti. "Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach", IEEE Transactions on Parallel and Distributed Systems, June 2001.
[21] Yasushi Saito, Brian N. Bershad, and Henry M. Levy. "An approximation-based load-balancing algorithm with admission control for cluster web servers with dynamic workloads", Journal of Supercomputing, vol.53, No.3, 2010, pp 440-463.

Wei Zhang HeBei Province, China. Birthdate: Dec, 1983. is a PhD candidate in the Department of Computer science and Technology at Beihang University. She received her master degree in 2008. Her research interests include virtualization, load balancing and cloud computing.

Huan Wang HuNan Province, China. Birthdate: Oct, 1986. is Computer Science and Engineering Master, graduated from Dept. Computer Science Beihang University. And research interests on operating system, load balancing, parallel computing and massive data processing.



Binbin Yu born in July 1987. Now study in Computer Science College at Beihang University for the master degree. Mainly concentrates on load balancing and cloud computing.

Wei Xu Fujian Province, China. Birthdate: Nov, 1986. is Computer Science and Technology MA, graduated from Dept. Computer Science and Engineering BeiHang University. And research interests on virtualization, operating system, high performance computing, cloud computing.

Mingfa Zhu born in 1945. Ph.D, Professor, Senior membership of China Computer Federation. His main research areas are computer architecture, computer system software, high performance computing, virtualization and cloud computing

Limin Xiao born in 1970. Ph.D, Professor, Senior membership of China Computer Federation. His main research areas are computer architecture, computer system software, high performance computing, virtualization and cloud computing.

Li Ruan born in 1978. Ph.D, Lecturer, Membership of China Computer Federation, Her main research areas are computer architecture, computer system software, high performance computing, virtualization and cloud computing.