J Inf Process Syst, Vol.19, No.5, pp.702~712, October 2023
https://doi.org/10.3745/JIPS.04.0289

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Research on the Application of Load Balancing in Educational Administration System

Junrui Han[1] and Yongfei Ye[2,*]

Abstract

Load balancing plays a crucial role in ensuring the stable operation of information management systems during periods of high user access requests; therefore, load balancing approaches should be reasonably selected. Moreover, appropriate load balancing techniques could also result in an appropriate allocation of system resources, improved system service, and economic benefits. Nginx is one of the most widely used load-balancing software packages, and its deployment is representative of load-balancing application research. This study introduces Nginx into an educational administration system, builds a server cluster, and compares and sets the optimal cluster working strategy based on the characteristics of the system, Furthermore, it increases the stability of the system when user access is highly concurrent and uses the Nginx reverse proxy service function to improve the cluster's ability to resist illegal attacks. Finally, through concurrent access verification, the system cluster construction becomes stable and reliable, which significantly improves the performance of the information system service. This research could inform the selection and application of load-balancing software in information system services.

Keywords

Load Balancing, Session Sharing, Educational Administration, Reverse Proxy

1. Introduction

The continuous development of the Internet has led to an increase in Internet users. Although network services have enhanced efficiency and convenience in various aspects of management, various challenges persist. With the rapid growth in the number of Internet users, website traffic has increased exponentially. Moreover, certain events and times are associated with high website traffic. Additionally, some business features also result in users accessing the business servers simultaneously, often causing a performance bottleneck in server response.

Network service providers typically select a cluster strategy based on multiple servers to improve the working ability of web servers. Numerous concurrent access requests from users are first distributed to different servers through load balancers, which ensure consistency in the processing of the user requests. This method effectively reduces the computing intensity and data flow of a single server, and solves the

Manuscript received Apil 18, 2022; first revision December 26, 2022; second revision February 22, 2023; third revision July 10, 2023; accepted July 13, 2023.

* **Corresponding Author:** Yongfei Ye (yeyongfei005@126.com)
[1] Office of Educational Administration, Hebei North University, Zhangjiakou, China (junruihan@163.com)
[2] School of Information Science and Engineering, Hebei North University, Zhangjiakou, China (yeyongfei005@126.com)

problem of a single-machine architecture not meeting high concurrency requirements. The choice of load balancer depends on the specific usage scenario of the user. Load balancers have different working principles and functionalities; however, they all comprise hardware and software. Hardware equipment includes products from enterprises such as F5, CISCO, and Radware, which have high-performance load-balancing capabilities and also provide solutions and optimization methods for security, disaster recovery, network monitoring, and other aspects. Hardware equipment exhibits satisfactory load-balancing performance, high efficiency, and good stability; its main deficiency is its high cost. The software equipment included a Linux virtual server (LVS), Nginx, and HAProxy. Users can configure load-balancing policies according to their requirements at a low cost. Compared with hardware equipment, software equipment can be both economical and practical, and its disadvantage is that it requires a self-optimized configuration; however, there are relatively few practical application cases at present. Therefore, the representative Nginx software is selected as the research object to explore strategy selection and optimization in the specific application process to provide further experience for the practical application of the software equipment.

This study introduces Nginx into the educational administration system, compares and selects the best Nginx technology strategy, and investigates the optimal configuration of server cluster load balancing at the peak of user access according to the performance of Nginx and the network architecture of the educational administration system.

2. Nginx

Nginx is an open-source lightweight web service that provides high-performance HTTP/HTTPS [1]. Because of its stability, low resource consumption, high performance, and other characteristics, it has been widely favored by mainstream network service providers in recent years [2].

Nginx can solve the problem of server load balancing under high concurrency of user requests. Its advantages include high concurrency, open-source code, and very efficient static content processing; therefore, it is often used as a load balancer for external services on websites [3]. Moreover, Nginx consumes fewer resources when a server runs. After optimization, the number of concurrent connection responses can reach 30,000, which effectively addresses the C10K problem [4].

As a server, Nginx's value lies not only in load balancing but also in its reverse agent function [5]. Servers in the server cluster are prone to server load pressure imbalance while responding to requests, which not only underutilizes the server resources, but also leads to a decline in the overall performance of the cluster system [6]. In such cases, Nginx can provide load allocation strategies and feed Web server response information back to the requesting users, serving as a reverse proxy. A reverse proxy load-balancing diagram is shown in Fig. 1.

When a user initiates an access request, Nginx allocates the web servers and forwards a response from them. In this state, the server IP is hidden [7], and the client cannot obtain specific web server information. This effectively prevents an external network from maliciously attempting to attack the cluster server maliciously [8], guarantees the normal service function of the system, adds security protection at the front end of the system, and provides a barrier to system security.

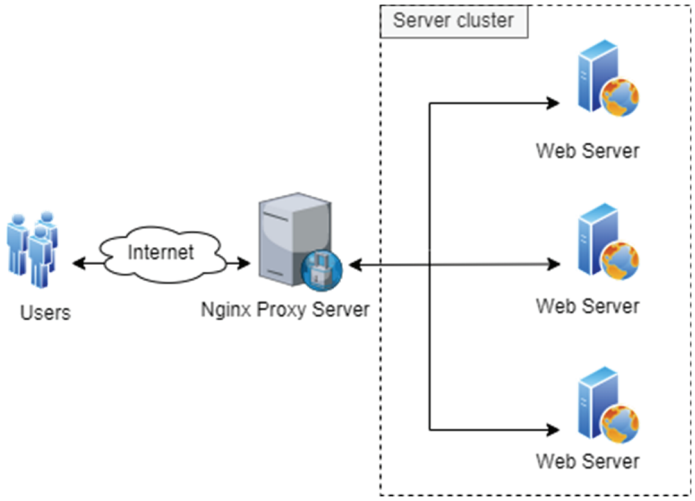

Fig. 1. Architecture of the Nginx reverse proxy.

2.1 Nginx Working Strategy

To maximize the performance of the cluster system, when users send requests to web servers, Nginx is typically used to set an appropriate balance strategy to forward the user requests to the back-end node servers. Subsequently, multiple servers in the cluster share network requests, and the server nodes work evenly, which expands the server bandwidth, reduces the average network response time, and improves network flexibility. Redundancy is also provided to ensure that businesses can still run normally even when their servers are unavailable [9]. When a server node in the cluster fails, Nginx can create the failed node offline by detecting the node status to ensure high availability of the server node in the cluster system.

The Nginx load-balancing function is mainly realized through its built-in load-balancing strategy, which includes the following strategies [10].

1) Polling strategy

The node servers in the cluster have the same response level, and the system assigns client requests to different servers in the cluster for processing according to the time order. If a server fails, the Nginx service will automatically take the server node offline. This is the default policy applicable to a situation in which the configuration of the servers in the cluster has little difference, and the service request remains stateless.

2) Weight strategy

The node server task assignment in the cluster is performed using Nginx according to weights. The default value of each weight is 1, and Nginx usually assigns user requests to the corresponding node servers in the order of weight value. A large weight indicates a higher allocation probability. This policy is suitable for situations in which the hardware configurations of the node servers in the cluster are very different.

3) IP-hash strategy

Nginx uses the client IP address for hash calculation and allocates the corresponding Web server in the cluster to link with the user according to the hash function value of the IP address. This approach ensures that the same IP address is assigned to the same Web server, guaranteeing session continuity and resolving issues associated with sessions spanning multiple servers. This strategy also applies to state services.

Nginx can act as a front-cache server, caching front-end requests and improving the performance of web servers. This creates a local copy of the content that the user has recently accessed. When the data are accessed again within a period of time, Nginx does not need to make a request to the backend, thereby reducing network congestion and data transmission delay, and improving user access speed. Nginx can also be used as a forward proxy server. The client initiates a request from the target server. After the target server responds, the request is returned to the client through the Nginx forward agent, which effectively protects the data security and personal privacy of the client.

3. Characteristics of the Educational Administration System and Analysis of Nginx Deployment

3.1 Characteristics of the Educational Administration System

The system network architecture operates in a B/S mode. Once users are assigned roles by the system, they can log into the system through authentication and initiate requests to the Web server through a browser. After authentication, the server generates a client session ID and returns it to the client, who then receives the session ID and saves it in a cookie. The client visits the server again and provides a session ID. When the server receives a request from a client, it first checks for the session ID. If it does not exist, the server creates a new session ID and repeats the above process. Otherwise, the server goes through its session file, finds the session ID, and transfers the information. A client session flowchart is shown in Fig. 2.

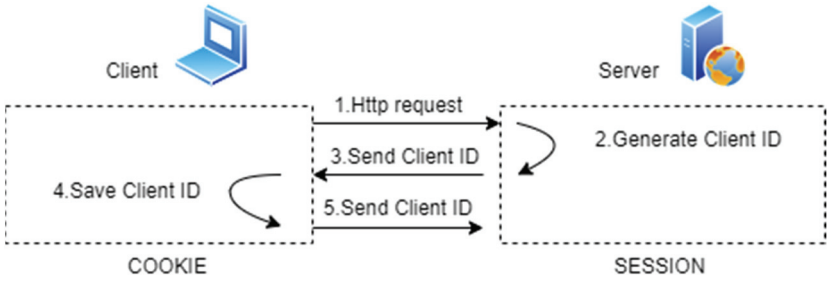

Fig. 2. Client session flowchart.

3.2 Analysis of Nginx Deployment

Establishing a server cluster ensures the service stability of the system during peak hours and satisfies the system session requirements. If the IP-hash strategy is used, users with the same IP address and fixed

node server in the cluster are identified through a hash calculation and its value, which meets the requirements of the educational administration system. However, this method suffers from the problem of access to one server from the same LAN user, and access aggregation occurs during peak hours, which seriously affects the server response time and system throughput under high concurrent requests.

The server cluster introduces Nginx to achieve load balancing, and the node servers in the cluster must meet session content synchronization, that is, session content sharing of the node servers in the cluster.

The remote dictionary server (Redis) service mode is introduced to facilitate session content sharing. Redis is a distributed cache system with key-value as storage objects. During its operation, all stored data are loaded into the memory, where the operation of the cache data is completed with extremely high response speed and throughput. Redis saves sessions, and node servers in the cluster share sessions through Redis distributed storage to ensure the normal completion of user requests. The system topology diagram is as shown in Fig. 3.

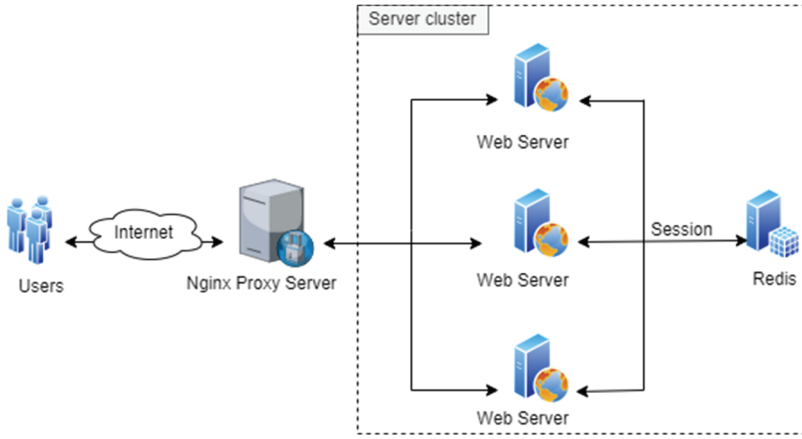

Fig. 3. Nginx plus the Redis load balancing mode.

4. Application of Nginx Deployment and Configuration

4.1 Web Server Configuration and Load Balancing Policy Settings

1) Web server configuration

In the Tomcat software of the web server, the following code is compiled in the conten.xml file of the Conf directory.

```
<Valve className="com.orangefunction.tomcat.redissessions.RedisSessionHandlerValve"/>
<Manager className="com.orangefunction.tomcat.redissessions.RedisSessionManager"
    host="x. x. x. x"
    port="6379"
    database="0"
    password="xxxxxx"
    maxInactiveInterval="1800"/>
```

Here, the host is the Redis server address, post is the Redis service port number, and the database is the number showing that Redis stores the session. MaxInactiveInterval indicates that the session expiration time is 1800 s.

The three files, commons-pool2-2.2.jar, jedis-2.7.2.jar, and tomcat-redis-session-manage-tomcat7.jar, are added to the lib directory in the Tomcat software.

2) The Nginx load-balancing policy settings

In the Nginx server, the upstream module of the nginx.conf file is called jwServer, the default service port of the web server in the cluster is 80, the maximum failure is set to three, and the failure timeout is set to 15 seconds. If there are three server requests, the failure number is 3, and the server is paused for 15 seconds.

```
upstream jwServer{
   server web1ip:80 max fails=3 fails timeout=15s;
   server web2ip:80 max fails=3 fails timeout=15s;
   server web3ip:80 max fails=3 fails timeout=15s;
                    }
```

Implementing a reverse proxy with proxy_pass in the location module forwards the customer request service to the defined server upstream.

```
location / {
   proxy_pass http://jwServer;
            }
```

In the configuration of the print server, according to the deployment characteristics of the educational administration system, a fixed web server is set up to process the print service. The print server location is set to cjPrint in Nginx, and its external access address is http://(webip).

```
location /cjPrint/{
   proxy_pass http://(webip)/;
                    }
```

4.2 Nginx Security Policy Settings

Users make access requests through the browser, and data transmission is completed through the HTTP protocol. If there is no encryption mechanism in the HTTP protocol, there are some deficiencies, such as no verification of customer responses and failure to identify camouflage. Nginx can rewrite the URL with the Rewrite module using the HTTPS protocol combined with SSL and HTTP, which has the following advantages: (1) enables encryption of data using the security features of the HTTPS protocol itself to prevent the leakage of sensitive data; (2) enables completion of the identity authentication problem of a third party to prevent the system from being accessed by other unauthorized personnel; and (3) prevents

malicious tampering of user data during network transmission to ensure data integrity [11]. After establishing a secure communication line with SSL, HTTP communication is conducted online to increase data transmission security.

The server module is configured in the nginx.conf. The Nginx listening port is 80. Server_name indicates the external access domain name, and rewrite is used to redirect the URL using the HTTPS transport protocol.

```
server {
   listen 80;
   server_name External domain name of educational administration system;
   rewrite ^(.*)$  https://$host$1 permanent;
      }
```

The SSL protocol and its configuration fields are shown in Table 1.

Table 1. SSL protocol configuration fields and definitions

Field name	Definition
access_log	Defining the Nginx log path and the record format
ssl_certificate	Setting the certificate file and path
ssl_certificate_key	Setting the private key and path of the associated certificate
ssl_session_cache	Optimizing HTTPS performance and access process
ssl_session_cache	Defining session sharing mode capacity
ssl_session_timeout	Defining session timeout
ssl_ciphers	Defining the encrypting method of server communication data
ssl_prefer_server_ciphers	(parameter is on) the client-side giving priority to SSLv3 and TLS cryptographic protocols

To prevent HTTP access from displaying error 400, configure port 443 for SSL and enable the simultaneous use of both the HTTP and HTTPS protocols.

The configuration content is as follows:

```
server {
   listen   443  ssl;
   server_name (external domain name of educational administration system);
access_log  /xx/log/xx/(external domain name of educational administration system).access.log main;
ssl_certificate /xx/nginx/xx/jw/1_(external domain name of educational administration system)_bundle.crt;
   ssl_certificate_key /xx/nginx/xx/2_(external domain name of educational administration system).key;
   ssl_session_cache  shared:SSL:1m; # mode is shared; size is 1 m
   ssl_session_timeout  6m; # ssl the session timeout time is 6m;
   ssl_ciphers  HIGH:!aNULL:!MD5; # MD5 algorithmic encryption;
   ssl_prefer_server_ciphers on;
        }
```

5. Experimental Verification

5.1 Web Client Virtual Machine Configuration

The experiment utilizes a system consisting of three servers with Linux operating systems, each with the following attributes: CPU, Intel Xeon E5-2630 V3; main frequency, 2.40 GHz; memory, 16 G; and hard disk, 300 G. One Redis server is also used.

5.2 Test Process

In the experimental setup, the operating system Windows 10 Professional Edition is used, Apache 2.4 is employed as the test software, and the experimental simulation environment involves users ranging from 1,000 to 10,000 in increments of 1,000. The objective of the experiment is to assess the performance variations of Nginx under high concurrent request conditions.

The command line is entered in the Windows PowerShell as an administrator:

```
./ab -n x -c y http://jwserver/path/login.html;
where -n x represents x requests being made at a time, and c y represents y users who concurrently access the server.  x> = y.
When the number of users is 1000, the system operation is as follows:
……
Benchmarking external domain name of educational administration system server (be patient)
Server Hostname (service host name):      Website domain name
Server Port: (server port)           80
……
Concurrency Level: (current concurrent volume)      1000 (1000 concurrent user requests)
Time taken for tests: (test completion time)   4.059 seconds
……
Failed requests: (number of request failures)       0
……
Requests per second: (requests processed per second)   246.34 [#/sec] (mean)
Time per request (user request time):       4059.366 [ms] (mean)
Time per request: (each request time)       4.059 [ms] (mean across all concurrent requests)
……
Percentage of the requests served within a certain time (ms)
50%   1410
……
100%   2929 (longest request)
```

5.3 Statistical Data

The results obtained from the tests are shown in Tables 2 and 3. The comparison results are shown in Fig. 4.

5.4 Experimental Results

The pressure test experiment reveals that, in the context of high user concurrency and no Nginx agents, the performance of the web server in the educational administration system fluctuates significantly.

During the stress test, within the range of 1,000–10,000 user requests, the error message "The timeout specified has expired" appears three times during the user concurrent request levels of 4,000, 7,000, and 10,000, indicating that the web server cannot fully guarantee the system access requirements in a high concurrency state. The system balances user requests to each web server in the server cluster composed of Nginx agents; the entire educational administration system has no errors when 1,000–10,000 users access the system, and the system is stable in the high concurrent access test.

The test experiment also reveals that the Nginx and node servers form a server cluster, and both the IP-hash policy and the Nginx plus Redis mode can meet the system requirements of client access, that is, maintain the session state. However, the IP-hash hash has the problem of aggregation with LAN access, and LAN proxy access is not applicable. Although the Nginx + Redis mode increases the number of Redis hardware servers, the configuration requirements are not high. Through an experimental comparison, the addition of the Redis server consumes less time than the IP hash, and the completion time is better than that of the IP hash strategy. When the high-concurrency state is within 10,000, the request time of the client fluctuates slightly. Therefore, it can be applied to educational administration systems, owing to its relatively stable performance.

Table 2. System response statistics table (IP hash strategy)

Concurrency (pcs)	Time taken for tests (s)	Requests per second	Time per request (ms)	Time to complete 50% (ms)
1,000	3.507	285.11	3.507	1,866
2,000	7.300	273.98	3.650	3,667
3,000	11.649	257.54	3.883	5,506
4,000	14.437	277.07	3.609	7,287
5,000	17.802	280.86	3.560	9,125
6,000	21.937	273.51	3.656	10,972
7,000	25.707	272.30	3.672	13,048
8,000	30.022	266.47	3.753	15,616
9,000	32.464	277.23	3.607	16,356
10,000	36.624	273.05	3.662	18,683

Table 3. System response statistics table (session sharing strategy)

Concurrency (pcs)	Time taken for tests (s)	Requests per second	Time per request (ms)	Time to complete 50% (ms)
1,000	4.059	246.34	4.059	1,410
2,000	8.286	241.38	4.143	2,680
3,000	12.703	236.17	4.234	4,114
4,000	16.304	245.34	4.076	8,133
5,000	20.406	245.02	4.081	9,145
6,000	24.582	244.08	4.097	8,367
7,000	28.638	244.43	4.091	9,292
8,000	32.997	242.44	4.125	13,803
9,000	37.762	238.33	4.196	12,958
10,000	41.031	243.72	4.103	15,500

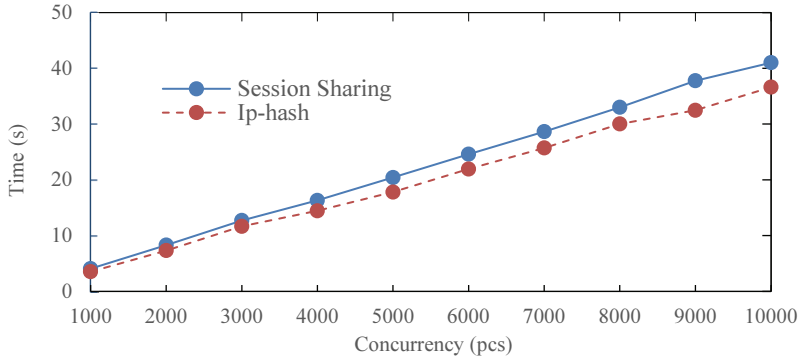

Fig. 4. Response time comparison of different Nginx strategies.

6. Conclusion

In big-data applications, the functions of information management systems are becoming increasingly perfect. With an increase in the number of users, the problem of system stability during peak system access must be addressed. The load balancing approach should leverage the technical advantages and analyze the characteristics of the system to meet specific needs. This approach ensures a rational allocation of system resources, appropriate application, and successful integration of technology with system requirements. Combining Nginx with a web server to build a server cluster can further optimize the service functions, ensure fast and stable access to the educational administration system, and solve the problem of an uneven system load when requests are highly concurrent to improve the stability of the system. Simultaneously, the Nginx reverse proxy function conceals the node server, preventing network attacks from accessing target information and improving the security of system services. This study focuses on the characteristics of the system, compares and analyzes different load-balancing strategies, and achieves an optimal configuration and reasonable application.

Acknowledgement

This paper is supported by Sports Science and Technology Research Project in Hebei Province of China (No. 2023QT08), Research on the Application of Blockchain Technology in College Student Status Management (No. 2221024B), the research and practice project of higher education teaching reform in Hebei Province of China (No. 2018GJJG313), Natural Science Foundation of Hebei Province of China (Grant No. H2022405021), and Key Scientific Research Project of Colleges and Universities in Hebei Province (No. SD2022048).

References

[1] R. Soni, “Nginx core architecture,” in *Nginx: From Beginner to Pro*. Berkeley, CA: Apress, 2016, pp. 97-106. https://doi.org/10.1007/978-1-4842-1656-9_5

[2] M. Pan and W. Li, “Research on Nginx load balancing algorithm based on ESXi performance counter,” *Journal of Jimei University (Natural Science)*, vol. 25, no. 1, pp. 76-80, 2020. https://doi.org/10.19715/j.jmuzr.2020.01.12

[3] J. Liu, Z. Fang, and B. Wang, “The improvement and implementation of the high concurrency web server based on nginx,” *Computing, Performance and Communication Systems*, vol. 1, no. 1, pp. 1-7, 2016. https://dx.doi.org/10.23977/cpcs.2016.11001

[4] D. Kegel, “The C10K problem,” 2014 [Online]. Available: http://www.kegel.com/c10k.html.

[5] L. Zhu, "Security deployment scheme of web system based on Nginx rechnology," *China Computer & Communication*, vol. 31, no. 17, pp. 172-176, 2019.

[6] S. M. Irteza, H. M. Bashir, T. Anwar, I. A. Qazi, and F. R. Dogar, “Efficient load balancing over asymmetric datacenter topologies,” *Computer Communications*, vol. 127, pp. 1-12, 2018. https://doi.org/10.1016/j.comcom.2018.05.010

[7] V. S. Randhe, A. B. Chougule, and D. Mukhopadhyay, “Reverse proxy framework using sanitization technique for intrusion prevention in database,” in *Proceedings of the 3rd International Conference on Computational Intelligence and Information Technology (CIIT)*, Mumbai, India, 2013, pp. 200-208. https://doi.org/10.1049/cp.2013.2592

[8] M. Chinipardaz and M. Noorhosseini, “A study on cell association in heterogeneous networks with joint load balancing and interference management,” *Telecommunication Systems*, vol. 66, pp. 55-74, 2017. https://doi.org/10.1007/s11235-016-0272-1

[9] D. Sharma, “Framework for achieving load balancing in web clusters based on load factor,” in *Proceedings of 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, Gurgaon, India, 2017, pp. 327-331. https://doi.org/10.1109/IC3TSN.2017.8284500

[10] Z. Liu, “Research on the construction of educational administration management system based on H3C CAS HA high reliability and Nginx load balancing,” *Technology Wind*, vol. 455, no. 15, pp. 85-86, 2021.

[11] Musiccoder, “HTTPS practice for large websites (1) – HTTPS protocol and principles,” 2015 [Online]. Available: https://blog.csdn.net/luocn99/article/details/45460673.

Junrui Han **https://orcid.org/0000-0002-0803-1130**

He received B.S. degree in the Department of computer science and technology from Beijing Institute of Information Engineering in 2002, and M.S. degree in computer application from Hebei University of Engineering in 2011. He is currently an engineer in the Office of Educational Administration, Hebei North University, Zhangjiakou, China. His research interests include information processing and privacy protection.

Yongfei Ye **https://orcid.org/0000-0002-1027-0501**

She is working as an associate professor in School of Information Science and Engineering in Hebei North University, Zhangjiakou, China. She received M.S. in computer software and theory from Yanshan University in 2008. Her research interest fields include information security, data analysis and precision agriculture.