Design and Implementation of Visualization System Based on Network Mass Data

Zihan Zhuo, Jianwu Dong, Yueying He
National Computer Network Emergency Response Technical Team/Coordination Center of China
Beijing 100029, China
e-mail: zzh@cert.org.cn

Lixuan Guo, Bin Peng
School of Computer Science and Engineering
Beihang University
Beijing 100191, China
e-mail: ever.guolx@outlook.com

***Abstract*—With the extensive applications of Internet, the Internet data have grown at full speed. To directly reveal the relationship among mass data and the latent characteristics in information with visualization techniques has become an important research topic. Concerning small resource storage capacity and low data exchange efficiency of the traditionally visualization C/S system, this paper adopts the computer distributed framework and the efficient network transmission technique to finish overall framework design of the network data visualization system, and realizes it based on the Web full-stack technique. Experimental results suggest that the system can provide comprehensive visualization services for mass data and contribute to extraction of hidden information efficiently and directly from data, thus facilitating decision formulation.**

Keywords—information visualization; Web system; network; mass data

I. Introduction

With the rapid development of Internet, mobile Internet and Internet of Things (IoT), the world has been ushered in an era of data explosion. From various management and operation data of commercial companies to socialized data of personal mobile terminals to mass information data of Internet, information is increasing quickly all the time. How to directly reveal the relationship among mass data and the characteristics hidden in information has become a global research issue, which leads to the emergence of a brand-new scientific field, called information visualization [1]

Mass data visualization technique combines multiple theories and methods, including scientific visualization, man-machine interaction, data mining, image graphics and cognitive science. It is mainly applied to revealing hidden knowledge of abstract information without geometric attributes [2, 3]. Visualization transforms data information and knowledge into a visual form, and makes full use of people's natural capability of fast recognition of visual models [4, 5]. The modern society is teeming with information. Research, application and development of visualization techniques have fundamentally changed the way how large-scale complex data are understood, thus leading humans to get new insights and formulate efficient decisions.

Current visualization system framework can be mainly divided into two models, namely Client/Server (C/S) and Browser/Server (B/S) [6]. The C/S visualization system framework has a higher requirement of the rendering capacity of the computer hardware. But its platform compatibility is poor, so client software on different platforms need to be developed [7]. With the development of Web techniques, the B/S visualization system framework has found increasingly extensive applications in the big data visualization field. Users can operate in the browser, thus significantly lowering the threshold of client software and hardware and showing good compatibility [8-10]. This paper designs and realizes a network mass data visualization system based on the B/S framework, and focuses on discussing distributed storage framework and data visualization techniques.

II. System Design

A. System Framework

The overall framework of the large-scale network data visualization system is shown in Fig. 1. The system consists of three layers, including the database cluster, the database exchange layer and the data application layer.

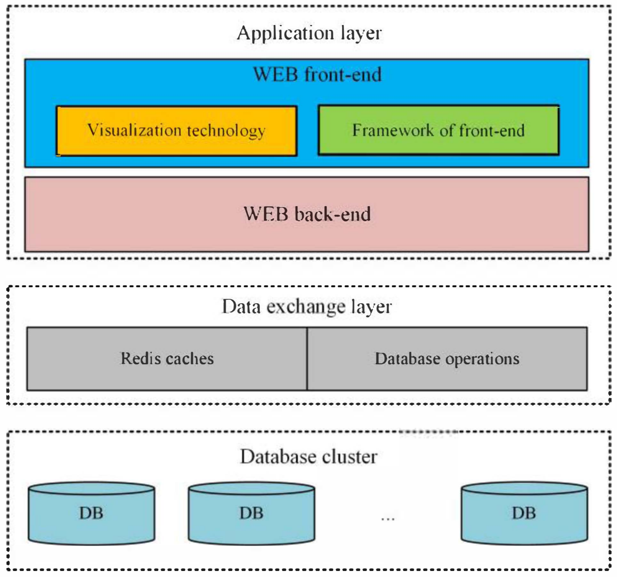


Figure 1. System framework.

The database layer provides data support for the system, which is used to store all the original data, the processed intermediate data and the final displayed data required by the system. Concerning the large scale of mass data, the storage capacity and the searching efficiency of the stand-alone database cannot meet requirements. It is thus necessary to build a database cluster to improve the system performance.

The data exchange layer packages the operation of getting access to the database, and provides the Redis buffer, which is used to buffer commonly-used data and improve the efficiency of the application layer to read data.

The application layer is a website, which can be divided into the back-end and front-end. The back-end is in charge of logic processing, database communication and exchange, data processing in the bottom layer. Besides, the back-end can provide user permission and system configuration. The front-end is used for user and system interaction, and data visual display. Moreover, it is responsible for logic handling of the front-end function and organizing data for rendering by the visualization tool set.

B. *Mass Data Storage Framework*

The network data are complex. Their estimation and prediction has a high requirement of high concurrency and reading-writing efficiency of the database in the TB/PB scale. However, traditionally mainstream storage schemes such as Oracle and MySQL are inapplicable to the big data distribution scenario [11, 12]. In order to facilitate structural optimization and distributed expansion of data, MongoDB non-relational database in the format of Key-Value is adopted for data storage. Compared with traditional relational databases, MongoDB has the following characteristics [13-16]:

1) It can store non-structuralized data. Even if in the same collection, data can have different formats, and are easy for expansion.

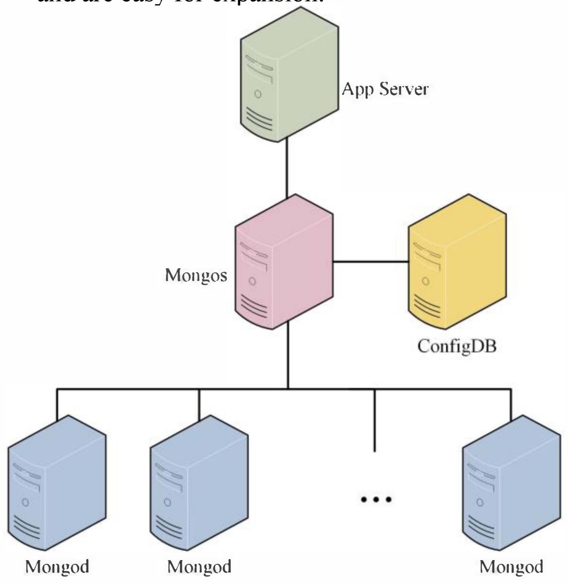


Figure 2. MongoDB cluster structure.

2) It does not support the structured query language (SQL) operation. Only the specific SQL of MongoDB can be used to access data. Currently, multiple programming languages have official links to support the MongoDB reading-writing operation.

3) It is easy for horizontal expansion. The horizontal fragmentation of the relation database calls for complex manual configuration. MongoDB can automatically set data fragments and conduct load balancing processing.

Since system data come from multiple sources, data from different sources have different structures. It is necessary to design corresponding storage models according to data structures. On the one hand, MongoDB can simplify the model construction. And on the other hand, It will not change existing data models when new data sources are increased, so that it is easy for expansion.

In view of the large-scale data collection, storage capacity and the reading-writing efficiency optimization is taken into consideration, and the MongoDB cluster shown in Figure 2 is built. Mongod is the storage node of the MongoDB cluster, which can be used to store data. ConfigDB server stores configuration information of different Mongod and Mongos. Mongos stores routing information of every Mongod, and is communicating with the App Server communicate. It should be noted that communication between Mongos and nodes within the cluster is transparent to the App Server.

C. *Efficient Data Exchange Framework*

The traditional back-end and front-end data exchange can be realized through a complete HTTP communication. However, the HTTP header is relatively huge. When resources required are relatively small, the efficiency of HTTP communication is extremity low because a large amount of bandwidth and server resources will be occupied.

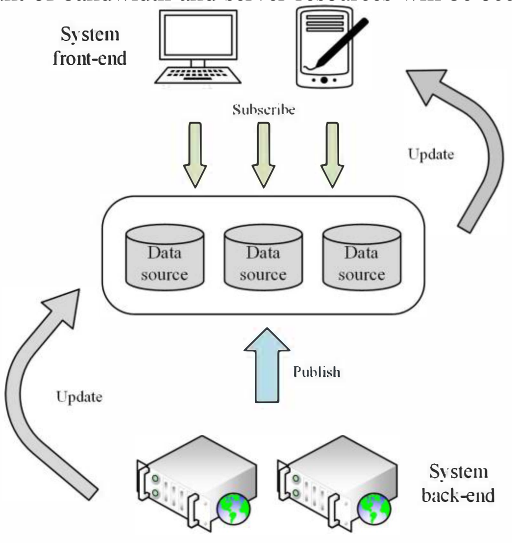


Figure 3. Data exchange technique.

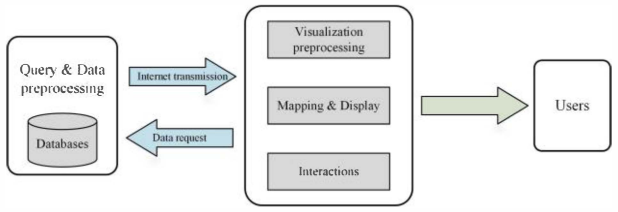


Figure 4. Visualization framework.

While the data size of a single visualization page is not huge, the data display should be updated without page refreshing under general conditions. If the front-end adopts the traditional JavaScript polling mode to keep on sending HTTP requests as stated above, the data exchange efficiency would be probably not a satisfactory user experience.

In order to improve the data exchange efficiency, the system adopts the WebSocket as the communication protocol. After one TCP connection, the WebSocket will conduct follow-up data transmission based on the connection. There is no need for it to send HTTP messages all the time, which can greatly improve the network transfer efficiency [17]. As shown in Fig. 3, the system adopts the "Publish /Subscribe" model for data acquisition. The back-end publishes multiple data sources and the front-end subscribes for corresponding data according to demands. When the back-end data sources are changed, the front-end page can get real-time monitoring of changes and immediately update its page display.

D. *Visualization Framework*

The visualization framework is designed in the Web front-end of the application layer. As Shown in Figure 4, It can mainly be divided into front-end data subscription and data visualization display. Data can be subscribed based on demands. Based on HTML5 language and cognition habits, the front-end loads the visualization tool set for rendering. Besides, the system provides the interactive interface, which can adjust parameters from the perspective of time and space, directionally subscribe and display data.

III. System Implementation

A. *Web Back-end Techniques*

This system uses Meteor framework [18], which is a full-stack Web technology namely a framework simultaneously responsible for the front-end and the back-end.

1) Database schema: The data storage of visualization system is realized through MongoDB. Its collection (namely tables in the relational database) does not have to define a schema of specific format. However, definition of schema can help clearly describe the data structure and facilitate cognition and cooperation of data.
2) Data exchange: The front-end and the back-end adopt "Subscribe/Publish" model for data communication. The back-end defines the data publishing interface, while the front-end can subscribe for data according to page demands, and then conduct visualization rendering.
3) Data preprocessing: Before data are published, the back-end extracts data from the corresponding collection. If data required are analytical or statistical results, it is necessary to conduct data preprocessing. Then, corresponding publishing operations are conducted and the front-end is responsible for data subscription.
4) Router system: Different uniform resource locators (URLs) are corresponding to different pages. The router system is responsible for building a logic relationship between URLs and pages. When specific URLs are visited, corresponding logics are implemented and correct pages are returned to users.

B. *Web Front-end Techniques*

The Meteor framework is built on the platform of Node.js [19]. Different from the commonly-used model-view-controller (MVC) framework, it does not transmit HTML on the Internet. The server just sends data while the client side is responsible for rendering, thus greatly alleviating burden of the server. During system development, maintenance or upgrade, change of system structures and patterns will be informed of the browser, the latter of which will refresh its pages and realize "hot push" of codes. During rendering big data, data changes of the server or the client side are automatically synchronized, which can improve efficiency of data exchange, computation and drawing to a large extent. The realization of the system front-end framework is shown in Fig. 5.

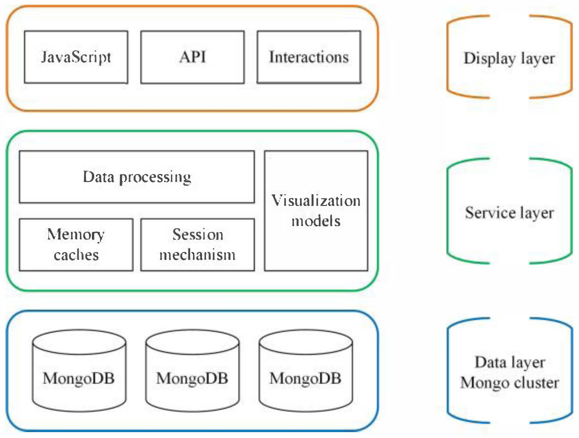


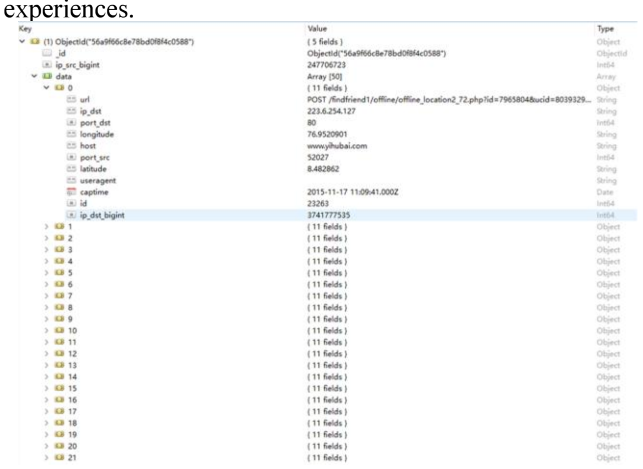
Figure 5. System front-end framework.

C. *Visualization Techniques*

Visualization techniques involved in this system include rendering and preprocessing module, mapping and display module and interaction module.

1) Rendering and preprocessing: Data processing is oriented towards visualization methods. The client side can conduct visualization transformation of data received (such as linear transformation, feature detection and smoothing) so as to reduce performance pressure of GPU hardware, accelerate imaging speed and improve the mapping efficiency of data on the displayer.
2) Mapping and display: Transform the numerical matrix into the image. Based on the visual cognition system of humans, proper graphs and charts are used to help get a more correct and direct understanding of data.
3) System interaction: Adjust data visualization effects according to input parameters or feedback information so as to display data characteristics in an all-around and multi-dimensional way.

In terms of large-scale network data, it is necessary to attach great importance to data preprocessing during real-time network data exchange. The quality of data preprocessing largely decides the comfortability of system experiences.



```
Key                                          Value                                                              Type
⌄ (1) ObjectId("56a9f66c8e78bd0f8f4c0588")   ( 5 fields )                                                       Object
    _id                                      ObjectId("56a9f66c8e78bd0f8f4c0588")                               ObjectId
    ip_src_bigint                            247706723                                                          Int64
  ⌄ data                                     Array [50]                                                         Array
    ⌄ 0                                      ( 11 fields )                                                      Object
        url                                  POST /findfriend1/offline/offline_location2_72.php?id=7965804&ucid=8039329...  String
        ip_dst                               223.6.254.127                                                      String
        port_dst                             80                                                                 Int64
        longitude                            76.9520901                                                         String
        host                                 www.yihubai.com                                                    String
        port_src                             52027                                                              Int64
        latitude                             8.482862                                                           String
        useragent                                                                                               String
        captime                              2015-11-17 11:09:41.000Z                                           Date
        id                                   23263                                                              Int64
        ip_dst_bigint                        3741777535                                                         Int64
    › 1                                      ( 11 fields )                                                      Object
    › 2                                      ( 11 fields )                                                      Object
    › 3                                      ( 11 fields )                                                      Object
    › 4                                      ( 11 fields )                                                      Object
    › 5                                      ( 11 fields )                                                      Object
    › 6                                      ( 11 fields )                                                      Object
    › 7                                      ( 11 fields )                                                      Object
    › 8                                      ( 11 fields )                                                      Object
    › 9                                      ( 11 fields )                                                      Object
    › 10                                     ( 11 fields )                                                      Object
    › 11                                     ( 11 fields )                                                      Object
    › 12                                     ( 11 fields )                                                      Object
    › 13                                     ( 11 fields )                                                      Object
    › 14                                     ( 11 fields )                                                      Object
    › 15                                     ( 11 fields )                                                      Object
    › 16                                     ( 11 fields )                                                      Object
    › 17                                     ( 11 fields )                                                      Object
    › 18                                     ( 11 fields )                                                      Object
    › 19                                     ( 11 fields )                                                      Object
    › 20                                     ( 11 fields )                                                      Object
    › 21                                     ( 11 fields )                                                      Object
```

Figure 6. System front-end framework.

IV. Results and Discussions

A. Data Storage Scale

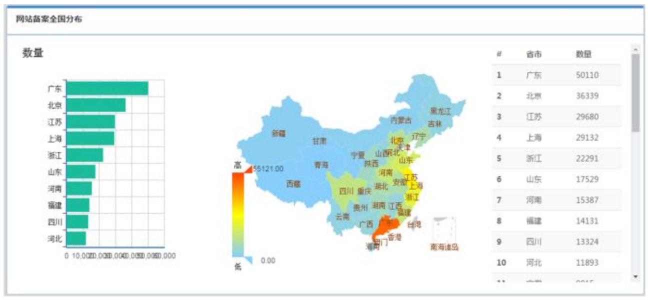
Compared with relational databases, MongoDB has a simple fragmentation configuration, which is easier for horizontal expansion [20]. To network data of a massive scale, reliability of data storage is of vital importance. The storage transcript often has to be set, such as one host node and one slave node or one host node and N slave nodes. When the host node encounters faults, users can quickly shift to the slave node to continue getting services and prevent data loss. The transcript set configuration of MongoDB is relatively simple. Nodes within the transcript set are mutually backups to achieve the host-slave backup.

All system data come from basic network data resources, including the IP address attribute knowledge base, network security events and network traffic logs. Fig. 6 is a data structure demonstration of one network traffic log. Every data occupies about 50 KB of the disk space on average. Currently, the traffic logs have been hundreds of millions, thus occupying the TB-level area of the disk space. Apart from dynamic flow logs, other static data have reached a scale of higher than 500 GB of the disk space. In order to store these data, ten sets of servers are adopted to build the database cluster. Every zone adopts the style of one host and one slave for deployment. When the data size increases to be larger than the current storage space, the space can be horizontally expanded through increase of the transcript set to safeguard system performance.

B. Visualization Effect

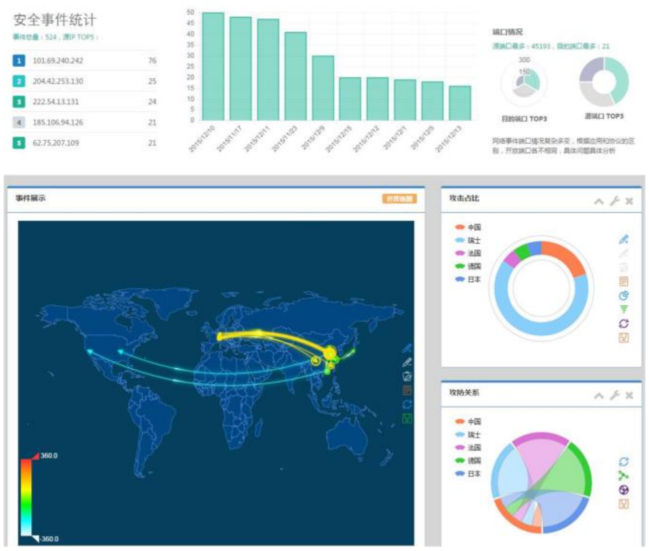
Based on the large-scale network data, the system can efficiently realize data exchange and visualization. The system can conduct statistical analysis of the basic data on the Internet, such as global IP address distribution, global website and domain-name registration, global Web server distribution, etc. As shown in Figure 7, based on the global website registration data, the visualization results are given. Fig. 7 could indicate that the information hidden in the big data is clearly and directly conveyed. For example, the regional development gap of China's Internet industry is huge, in that, advantages of Central and East China far exceed those of West and Northeast China.

From Fig. 8, it can be seen that a global monitoring and pre-warning model is built for network large-scale security events, which can contribute to direct discovery of potential safety dangers in the cyberspace. Experimental results suggest that the system front-end framework can meet different demands of data exchange and processing, and that multiple kinds of visualization tool sets can efficiently support rendering of multi-source big data.



#	省市	数量
1	广东	50110
2	北京	36339
3	江苏	29680
4	上海	29132
5	浙江	22291
6	山东	17529
7	河南	15387
8	福建	14131
9	四川	13324
10	河北	11893

Figure 7. Statistics of basic Internet data.



1	101.69.240.242	76
2	204.42.253.130	25
3	222.54.13.131	24
4	185.106.94.126	21
5	62.75.207.109	21

Figure 8. Monitoring of network security events.

V. Conclusions

The 21st century is an era teeming with an unprecedented quantity of information that humans are living among it. It is necessary to find an efficient way to help people discover models and knowledge hidden in complex information. This is the significance of information visualization.

Concerning defects of the current Web visualization system in terms of resource sharing, interconnection rebuilding and exchange capacity, the distributed storage framework and the efficient network data exchange technique are adopted to design the overall service and functional framework of the visualization system. According to the Web full-stack technique, a set of visualization system based on the network mass data is realized. Results suggest that the system can realize efficient utilization of data storage, computation and drawing resources on the Internet, provide all-around visualization services for data and contribute to efficient and direct extraction of useful information from the big data. The useful information thus extracted can facilitate decision-making.

References

[1] F. Liu, “Information visualization technology and application,” Ph.D. dissertation, Zhejiang University, 2013.

[2] D. Keim, H. Qu, and K.-L. Ma, “Big-data visualization,” Computer Graphics and Applications, IEEE, vol. 33, no. 4, pp. 20–21, 2013.

[3] N. T. Tam and I. Song, “Big data visualization,” in Information Science and Applications (ICISA) 2016. Springer, 2016, pp. 399–408.

[4] S. Sagiroglu and D. Sinanc, “Big data: A review,” in 2013 International Conference on Collaboration Technologies and Systems (CTS). IEEE, 2013, pp. 42–47.

[5] E. Y. Gorodov and V. V. Gubarev, “Analytical review of data visualization methods in application to big data,” Journal of Electrical and Computer Engineering, vol. 2013, p. 22, 2013.

[6] X. Zeng, “Brief analysis of b/s and c/s structure development and application,” Computer Knowledge and Technology, vol. 8, pp. 407–408, 2007.

[7] D. He, “An efficient remote user authentication and key agreement protocol for mobile client–server environment from pairings,” Ad Hoc Networks, vol. 10, no. 6, pp. 1009–1016, 2012.

[8] C. Zhang, T. Luo, and X. Wang, “Design and implementation of a web-based information visualization system,” Computer Systems & Applications, no. 12, 2009.

[9] W. Wang and J. Li, “The design of an open laboratory information management system based upon a browser/server (b/s) architecture,” World Transactions on Engineering and Technology Education, vol. 11, no. 1, pp. 41–45, 2013.

[10] H. Fang, Y. Jing, G. Han, and Y. Li, “Design and implementation of a fishery science management information system based on client/server and browser/server models,” African Journal of Agricultural, vol. 8, no. 18, pp. 1847–1851, 2013.

[11] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, “The rise of ”big data” on cloud computing: Review and open research issues,” Information Systems, vol. 47, pp. 98–115, 2015.

[12] S. Madden, “From databases to big data,” IEEE Internet Computing, no. 3, pp. 4–6, 2012.

[13] K. Chodorow, MongoDB: the definitive guide. ” O’Reilly Media, Inc.”, 2013.

[14] C. Gyorodi, R. Gyorodi, G. Pecherle, and A. Olah, “A comparative study: Mongodb vs. mysql,” in Engineering of Modern Electric Systems (EMES), 2015 13th International Conference on. IEEE, 2015, pp. 1–6.

[15] A. Boicea, F. Radulescu, and L. I. Agapin, “Mongodb vs oracle–database comparison,” in 2012 Third International Conference on Emerging Intelligent Data and Web Technologies. IEEE, 2012, pp. 330–335.

[16] Z. Parker, S. Poe, and S. V. Vrbsky, “Comparing nosql mongodb to an sql db,” in Proceedings of the 51st ACM Southeast Conference. ACM, 2013, p. 5.

[17] V. Pimentel and B. G. Nickerson, “Communicating and displaying realtime data with websocket,” IEEE Internet Computing, vol. 16, no. 4, pp. 45–53, 2012.

[18] J. Robinson, A. Gray, and D. Titarenco, “Getting started with meteor,” in Introducing Meteor. Springer, 2015, pp. 27–41.

[19] I. Strack, Getting Started with Meteor. js JavaScript Framework. Packt Publishing Ltd, 2015.

[20] C.-W. Huang, W.-H. Hu, C. C. Shih, B.-T. Lin, and C.-W. Cheng, “The improvement of auto-scaling mechanism for distributed database-a case study for mongodb.” in APNOMS, 2013, pp. 1–3.