*Journal of Data Analysis and Information Processing*, 2014, 2, 6-11
Published Online February 2014 (http://www.scirp.org/journal/jdaip)
http://dx.doi.org/10.4236/jdaip.2014.21002

# A Web Server Cluster Solution Based on Twitter Storm

**Xiaoling Xia, Lin Tian***
School of Computer Science and Technology, Donghua University, Shanghai, China
Email: *lin-95@qq.com

Received November 18, 2013; revised December 20, 2013; accepted January 27, 2014



## ABSTRACT

**Single Web server would become a bottleneck that influences the availability and stability of Web service. Ten years ago, what had been proposed is to add Web servers for resolving this problem—Web Server Cluster. In recent years, the concept of cloud computing has got rapid development, and is becoming the future development trend of the IT industry. One of the characteristics of cloud computing is putting lots of computing resources together to provide users with a unified service. In this paper, we have proposed a new Cloud-Based Web Server Cluster Solution, based on the existing cloud computing model—Twitter Storm. It involves a new way to handle the web request from client and some other new features compared to the traditional Web Server Cluster. Combining with cloud computing, it would be the new trend of Web Server Cluster, and its feasibility is described in the paper too.**

## KEYWORDS

**Twitter Storm; Web Server Cluster; Cloud Computing**

## 1. Introduction

The architecture of traditional Web Server Cluster is shown in **Figure 1**, which is currently the most widely used Web cluster approach. The front end is a Load Balancer, which is responsible for receiving requests from the client, forwarding request and scheduling according to the load of each server in the cluster. And the Web servers in the cluster deal with the request from the front end and return the action result. At the same time, Web servers can get data or information from database server by Local Area Network (LAN) [1]. In addition, due to the Web servers in the cluster running Web application independently, the state information in Web service, such as the Session information cannot be shared within the cluster. So there should be a Session Storage Server for sharing the state information between the Web servers [2].

The Web server in traditional Web Server Cluster was running independently, which means that there are Web application instances running on each Web server container [3]. In the meantime, the solution is in response to peak load, and the sever utilization rate is actually not high, generally about 5% - 20% [4].

According to the concept of Platform as a Service (PaaS) in cloud service, Xin Zhao, *et al*. [3] proposed a shared cluster topology structure based on process isolation—supports for multiple Web services sharing the same physical server cluster—named Shared WAS Cluster. The core idea is integrating multiple Web applications on the same physical server cluster by splitting, merging and migrating the instances of Web application which is running on the cluster. By this way, we can achieve a much higher server utilization rate. However, there are actually multiple instances running on the cluster for a single Web application service at the same time, which is not compatible to the cloud computing model [5]. The specified Web application service for users on the physical server cluster is supposed to have only one corresponding instance running on the cluster.

We presents a new Cloud-Based Web Server Cluster that addresses the problem faced by traditional Web Sever Cluster and also can guarantee the physical server utilization rate for supporting PaaS. All in all, the new solution must handle the following issues:

1) The bottleneck problem caused by single Web ser-

*Corresponding author.

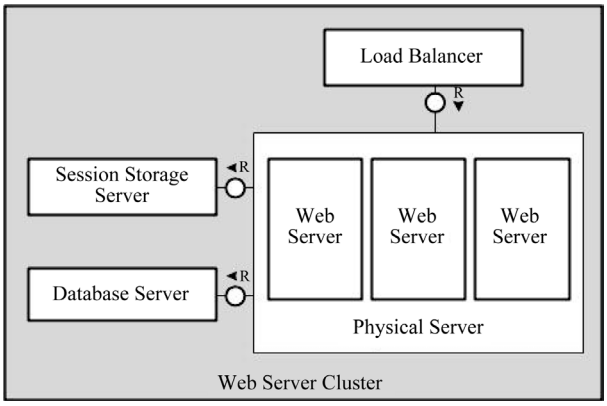


**Figure 1. Architecture of traditional web server cluster.**

ver, which means how to provide the same Web application service by more than one service entrances;

2) How to deal with the load balance and share state information between multiple physical servers;

3) How to support the cloud computing model of PaaS—there are multiple Web services on the same cluster, and there is only one instance for a Web service.

The structure of the paper is as follow: Section 2 presents the new Cloud-Based Web Server Cluster Solution from the aspect of architecture and physical deployment. Section 3 introduces Twitter Storm form the core idea of calculation model, cluster deployment and work flow. In Section 4, we discuss about how to solve the issues to the new solution with Twitter Storm. Finally, the paper concludes the work and presents our future research plan.

## 2. Cloud-Based Web Server Cluster

The architecture of the Cloud-based Web Server Cluster is different from the traditional Web Server Cluster, which is shown in **Figure 2**.

In the new solution, we use a unified cloud cluster instead of multiply independent Web servers, to handle the REST or RPC request [6] based on the HTTP from the client. These requests would be evenly distributed through the DNS server to the request node, which is responsible for data transmission with client. While the data processing work of the request can be forward to the other node in the cluster. All nodes in the cluster could retrieve data from the database server by the LAN and other ways.

The management of the node in the cluster is similar to the physical topology of Hadoop [7], which involved a master node and several slave nodes, and the master node coordinates with slave nodes rely on Zookeeper cluster. Meanwhile, the cluster take the distribution mechanism similar to the JobTracker [8] in Hadoop: once a data processing task of requests from client is received by the cluster, the master node would be responsible for assigning the task to one of slave nodes. And slave nodes should monitor the task from the master node and finish the related work of the task. To ensure the reliability of the cluster, the master node should have the ability of monitoring the execution of the task, and could reassign the task to another slave node in the cluster when the execution of one task is failed on previous slave node.

Compared to traditional Web Server Cluster, the Cloud-Based Web Server Cluster could have the following characteristics:

1) The physical servers can be seen as a unified cluster, which actually divides the servers into nodes for data transmission or data processing. This design makes it compatible with the PaaS, so the cluster can provide multiply Web service with multiply entrance for request while is acting just like a single Web server.

2) The request node is responsible for data transmission, while the data processing work is distributed by the master node and assured to be finished. It may not necessary to consider the load condition anymore, so we could use the DNS server for distributing the request from the client to the request node in the cluster.

3) Zookeeper provides a high-availability, high-performance coordination service, which can be used to store the state information [9]. It is usually used in the cloud computing model, such as Hadoop and Twitter Storm. We would apply the technique to the storage of state information for the new solution, which is specifically described in Chapter 4.

## 3. Twitter Storm

Twitter Storm is an open source distributed real-time computation system [9,10]. It is scalable, fault-tolerant, guarantees your data will be processed, and it has many use case: real-time analytics, online machine learning, continuous computation, distributed RPC, ETL, and more [10]. It has been used by many companies such as Twitter, Baidu, Alibaba, and more.

### 3.1. Core Idea of Twitter Storm

First, Let us introduce some concepts of Twitter Storm:

Stream: is the core abstraction in Twitter Storm. A stream is an unbounded sequence of tuples. And the tuples would be created and processed in a distributed and parallel way. Actually, one tuple is key-value pairs.

Spout: is the source of the stream, which means that it is the tuple producer. Spout takes responsibility for producing the origin tuple of the stream by retrieving data from file, database or other ways like Web request.

Bolt: is the compute unit of the stream, which consumes any number of input streams, does some processing, and possibly emit new streams. Bolts can do anything from run functions, filter tuples, do stream aggregation, do streaming joins, talk to database, and more.

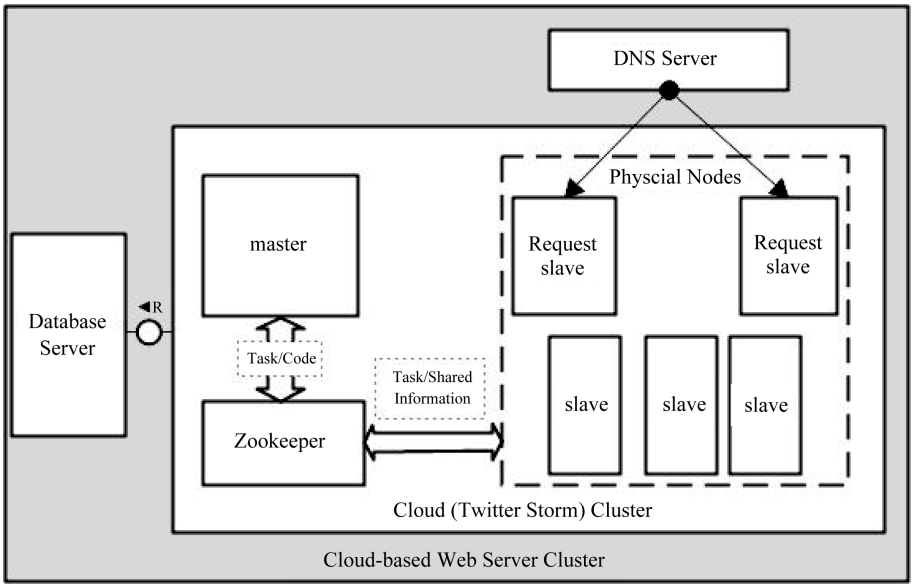


**Figure 2. Architecture of the cloud-based web server cluster.**

Topology: put the three concepts together, Spout produces the origin tuples, and emits the tuples into different streams based on specified requires. The tuples in the stream would be handled by different Bolts according to specified process logics. And then the result could be returned back if necessary. A complete set of real-time computing application could be handled by this way, which is abstracted as Topology in Twitter Storm. Topology is a real-time computing application logic. A simple structure of Topology is shown in **Figure 3**.

The arrow line between the nodes in the figure shows the flow of tuples. Once Spout produces a tuple, it would send the tuple to the Bolts that connected to the Spout. After received tuple from the Stream, Bolts can do tuple filtering, aggregation or other process logics, and then generate a new tuple passed to the next Bolt processing unit if necessary. A route from the Spout to a final Bolt, which would not generate any new tuples, is described as a Stream. All the elements in the figure make up a Topology, which means a Topology is made up of several Streams. From the figure, we would know that a Topology is a directed acyclic graph (DAG).

In short, a real-time computing application is abstracted as a DAG structure. The start point (Spout) receives data processing request from client. The request is mapped along the direction of the arrow line to be processed by the following nodes (Bolt). The real-time computing logic is finished as the action is done by the end point (Bolt).

## 3.2. Architecture of Twitter Storm Cluster

A Twitter Storm cluster is superficially similar to a Hadoop cluster, there are two kinds of node on the cluster:

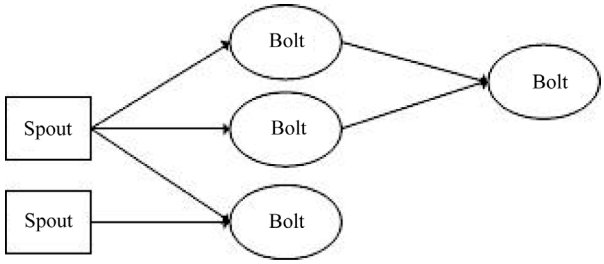


**Figure 3. A simple topology.**

the master node and the worker node. The master node runs a daemon called “Nimbus”, which is responsible for distributing code around the cluster, assigning tasks to machines, and monitoring for failures. Each worker node runs a daemon called the “Supervisor”, which listens for work assigned to its machine and starts and stops worker processes as necessary based on what Nimbus has assigned to it.

All coordinate between Nimbus and the Supervisor is done through a Zookeeper cluster. Additionally, the Nimbus and Supervisor are all-fast and stateless, all state is kept in Zookeeper or on local disk. This means you can kill Nimbus and Supervisors and they will start back up like nothing happened. This design leads to Twitter Storm being incredibly stable.

The deployment architecture of Twitter Storm has been shown in the **Figure 2**.

## 3.3. Handle Web Request with Twitter Storm

With the idea of Stream and Topology, Twitter Storm handles Web request through the following steps:

1) The worker node on which Spout is running receives the web request and emits the origin tuple.

2) The master node distributes the task to worker nodes according to the definition of Topology.

3) The worker nodes finish all the task origin from the request, and generate the result.

4) The worker node that emits origin tuple returns the result to the client.

So, multiple physical nodes in the cluster can process Web requests of Web service in parallel with the above idea. While in the Traditional Web Server, all the work associated with the specified Web request is finished by a single physical server, and the Web requests of Web service are handled by multiple threads concurrently, which is not suit for the cluster solution in the cloud Era.

DRPC has been implemented with this idea, providing the RPC service. The implementation shows that Web service can be realized on the cluster, and the idea of Stream and Topology in Twitter Storm creates the new way to implement real-time Web service.

## 4. Web Server Cluster Solution Based on Twitter Storm

Actually, Twitter Storm has implemented almost everything of the new Web Server Cluster Solution. While it only supports for RPC service base on SOA, does not support the REST service base on ROA, which means it cannot provide the human web service now. It is necessary to talk about the feasibility of the new Web Server Cluster solution.

### 4.1. Request & Response

With traditional Web server like Tomcat, the Web application in server side receives request from client, retrieves data or finishes process logic through function calls, and then sends the corresponding response to the client. The data flow of information was a strongly connected graph in the server. For example, SSH framework of J2EE, the request is received by the web layer, which calls the service layer for doing the process logic. The service layer retrieves the data by calling functions in the DAO layer. The data in server side goes back to the web layer along the way of the data persistence layer, the DAO layer, the service layer. Finally, the web layer makes the response and sends it to the client. And the problem is that we could make the independent Web servers into a unified cluster by the traditional way of callback function.

A Web application is actually a complex Twitter Storm Topology. A Web application is a strongly connected graph in traditional server, while the Topology of Twitter Storm is a DAG structure. With a little change to the core compute model of Twitter Storm, we could fix the difference between the two architectures. The modified compute model is shown in Figure 4.

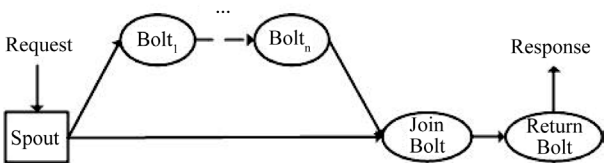


**Figure 4. The request & response procedure.**

The idea comes from the implement of DRPC in Twitter Storm. After Spout gets request from clients, it produces two kinds of tuple and sends to Blot1 and Join Bolt. The tuple in the process stream is executed through Blot1 to Blotn, and will be handled to Join Bolt with request information and process result finally. The Join Bolt stores the request information from the Spout into variables, and just waits the tuples from the processing stream. Once a tuple from processing stream is received, Join Bolt would make a tuple which will be sent to Return Bolt to assure that the request & response procedure is finished. Return Bolt is expected to return corresponding response to the actual request from the client.

The real-time Web service cannot be implemented in the cloud computing with the way of callback function. But Twitter Storm makes it possible to implements real-time Web service based on Server-Client architecture by the new way.

### 4.2. Load Balance

One of the main works in traditional Web Server Cluster is that how to coordinate the system of several stand-alone Web servers [11]. Because the Web servers are independent from each other, and each one of them has different processing capabilities, the load balancing for the cluster has been the key issue all the way. In the Web Server Cluster Solution based on Twitter Storm, the data processing is distributed to slave nodes by the master node, and the request will be guaranteed to be processed. There is no worry about that which node the request is processed on because all slave nodes is the same for the master node.

For this reason, Cloud-Based Web Server Cluster can handle the performance bottleneck problem caused by single Web server simply by the way: choosing several slave nodes in the cluster as request node (Spout) which is responsible for data transmission, and there is no need to consider about the complex load measurement. We could do the request distributing work by DNS server replaces for the Load Balancer. Meanwhile, the way of DNS server is much easier than the Load Balancer [12].

### 4.3. Storage of Sessions

One feature of the ROA is statelessness, which means that the HTTP request is complete isolated from others [6]. However, the Web server is expected to save the

information about the user who accesses the website from the time he enters to the time he leaves. As a consequence, the state information should be saved in the server to make the stateless HTTP request have state.

The Twitter Storm is not available for this requirement, so it only supports DRPC service based on SOA architecture for the moment. It does not mean that we could not do this work with Twitter Storm. As mentioned above, the design with Zookeeper cluster leads to Twitter Storm being incredibly stable. According to the idea, the Web Server Cluster Solution could provide a stateful Web service with the Zookeeper.

Zookeeper is thought to be a file system with high-availability without files or directories, and has the uniformed node, which called znode [7]. All of the znode in the Zookeeper constitute a tree structure begins with character "/". Each znode has its own properties and values. The state information is used for storing information of the client within a certain period, such as Session information. The two characteristics of state information are that it has several attributes and it need to be kept in a short time. Thus, we could store state information with Zookeeper as follows:

1) Create a child znode named SESSIONS for storage of the root node "/".

2) Generate a SESSION-ID for a HTTP request without SESSION-ID attribute, and store the associated information as the child node of SESSIONS with name of SESSION-ID. Assign an expire-time for the child node.

3) The HTTP request with SESSION-ID could get the state information by query Zookeeper with SESSION-ID.

4) Create a daemon (may be a Bolt) in the cluster, which can clean the SESSION-ID node that is expired.

As described above, the HTTP request with the same SESSION-ID could have same state information with the Zookeeper cluster in the Cloud-Based Web Server Cluster.

### 4.4. Support for PaaS

As one import form of cloud computing [4], Platform as a Service (PaaS) provides computing resources for Web application in the way of platform. PaaS is divided into the user layer, the application layer, the resource layer, the physical layer and the manage layer [13]. The Web Server Cluster based on Twitter Storm has new improvement in the resource layer: the Web server container is no longer independent, but a unified Twitter Storm Cluster. The Twitter Storm is actually a Web server container now, and the Web application is Topology.

As introduced in the Chapter 3, Twitter Storm is a real-time computing system with several Topologies running on it. The Web Server Cluster Solution based on Twitter Storm would support that several Web services provided by the same cluster and there is only one instance for a specified Web service is running based on the new architecture.

## 5. Conclusions

Focused on the bottleneck problem caused by single Web server, the paper proposes a new Web Server Cluster Solution based on cloud architecture. With the existing cloud computing model—Twitter Storm, we have introduced the solution in detail and discussed the feasibility of the architecture. For the trend of cloud computing, this new solution would be the best practice of the Web Server Cluster in the cloud Era.

In fact, Twitter Storm has implemented the RPC service based on Service-Oriented Architecture (SOA) while the REST service based on Resource-Oriented Architecture is not in its development plan. We take implement of the REST Web service [6] on the Twitter Storm cluster as our future work.

## REFERENCES

[1] T.-Y. Li, L. Xu and Z.-Q. Chang, "A New Solution of Web Cluster Based on Network Storage," *Application Research of Computing*, Vol. 20, No. 10, 2003, pp. 78-79, 112.

[2] H. Zhang, "Jee Web Cluster," InfoQ Enterprise Software Development Community, 2011. http://www.infoq.com/cn/minibooks/jee-webserver-cluster

[3] X. Zhao, W. Wang and W. B. Zhang, "Research on Resource Consolidation of Shared Web Application Server Cluster," *Journal of Frontiers of Computer Science and Technology*, Vol. 7, No. 1, 2013, pp. 25-34.

[4] M. Armbrust, A. Fox and R. Griffith, "A View of Cloud Computing," *Communications of the ACM*, Vol. 53, No. 4, 2012, pp. 50-58. http://dx.doi.org/10.1145/1721654.1721672

[5] K. Hwang, G. C. Fox and J. J. Dongarra, "Distributed and Cloud Computing from Parallel Processing to the Internet of Things," China Machine Press, Beijing, 2013.

[6] L. Richardson and S. Rudy, "Restful Web Service," Publishing House of Electronics Industry, Beijing, 2008.

[7] T. White, "Hadoop: The Definitive Guide," Tsinghua University Press, Beijing, 2012.

[8] C. Xu, H. Liu and L. Tan, "New Mechanism of Monitoring on Hadoop Cloud Platform," *Computer Science*, Vol. 40, No. 1, 2012, pp. 112-117.

[9] J. H. Zhao, "Study on Real-Time Data Processing Analysis Tools Based on Twitter Storm," *Shangqing*, Vol. 8, 2013, pp. 157, 274.

[10] N. Marz, "Storm Wiki," Storm, 2013. https://github.com/nathanmarz/storm/wiki

[11] Y. Pan, C. Gu and Z. H. Liu, "Study on the Application of Structure of Web-Server Cluster and Real-Time Scheduling Algorithms for Multiple Tasks," *Computer Measurement & Control*, Vol. 12, No. 1, 2004, pp. 74-76.

[12] X. Guo, "Building High Performance Web," Publishing House of Electronics Industry, Beijing, 2009.

[13] P. Xu, S. Chen and S. Su, "Architecture of PaaS for Internet Applications," *Journal of Beijing University of Posts and Telecommunications*, Vol. 35, No. 1, 2012, pp. 120-124.