Dynamic Load Balancing Method for Apache Flume Log Processing

UnGyu Han¹ and Jinho Ahn¹,
¹ Dept. of Comp. Scie., Kyonggi Univ., Iuidong, Yeongtong, Suwon 443-760 Gyeonggi, Republic of Korea
{nick_hug, jhahn}@kgu.ac.kr

Abstract. Recently, as web log data their users leave daily rapidly emerge as valuable assets for web service companies, designing log aggregator for collecting these log data in an efficient manner is also getting high attention in big data analytics research. However, we observe a representative data aggregator, Apache Flume, used for this purpose has some drawbacks on evenly distributing incoming log workload on collector agents. In this paper, we propose a new load balancing method to overcome this limitation in terms of the collected performance. The proposed method considerably alleviates the additional overhead incurred by the task migration and makes the load of the entire system as fair as possible by selecting the optimal task migration destination depending on the current load-state values of collector agents unlike the previous round-robin and random ones.

Keywords: Apache Flume, Log Aggregator, Log Collector, Load Balancing

1 Introduction

Flume is distributed from the Apache and Cloudera Foundation as one of open source projects included in the Hadoop project having the advantage that any developer can freely modify it. It allows each agent as a logical node consisting of source and sink components to generate and collect data. It is easy to install and use, and also can take a desired structure according to the corresponding user's control. However, Flume is load balanced for allocating the entire workload on agents according to the load of each agent node by applying the basic method that completely depends on the user-specified threshold. Therefore, a user has to specify a threshold considering data processing capability of the node. Also, the threshold is not automatically determined depending on the queue lengths of collector agents even if data processing capabilities of the node decreases, so there is inconvenience that the user has to manually fix it. The dispersion process forces the overloaded node to transmit excessive data over a user-specified threshold value to the other nodes. The disadvantage of this process results in performance degradation of overloaded node getting worse because it should also be in charge of this data transferring job.

In this paper, we propose a load management method that works with the Flume together to compensate for the disadvantages of Flume mentioned above. The proposed method automatically provide the system with more accurate thresholds by dynamically configuring them according to the current system performance unlike the previous one. Based on these thresholds, it makes some log producing nodes attached to each highly loaded node reconnected to under-loaded nodes to minimize the number of unevenly loaded collection
nodes. It performs a dynamic load-balancing to make the system environment able to handle incoming log workload as more as possible with the same set of collection agents compared with the existing one.

2 Problem of the Flume

In this section, we identify which kinds of problems the existing method of the current Flume has in detail. Each collection node may face some performance problem when data processing capability of the node is unexpectedly exceeded due to the enormous amount of log workload suddenly coming from data generating agents connected to itself. On the contrary, if the amount of data transmitted to the collection node is too small compared with its data processing capacity, the node may become under-utilized, even remaining almost in the idle state.

Fig.1 shows an example that overloaded and under-loaded nodes have occurred. In this case, the existing method forces overloaded collection node to transmit some excessive data the node keeps in its queue to another in either the round-robin or random way.

Fig. 1. An example of overloaded and under-loaded node occurrence in the Flume

This problem of the existing method fundamentally results from not considering the load condition of the task taking collection node. Therefore, when the method attempts to load balance the entire system, its performance may significantly varies depending on the state of the receiving collection node. There is also a problem that the method is
entirely dependent on his or her specified threshold. If its value is set too high, the method is infrequently invoked even if there are several overloaded nodes. Otherwise, the opposite behavior may occur. Therefore, the Flume requires an effective load balancing method to be able to adapt to dynamic characteristics of incoming workload.

### 3 The proposed protocol

When a collection node is determined as overloaded node, our proposed load balancing method enables some among data generating nodes connected to the overloaded collection node to be reconnected to the most suitable low-loaded node. In here, which node is overloaded or under-loaded is determined based on both the increasing rate of incoming log information in the queue of each collector agent and its occupancy rate like in Fig. 2. Also, it can automatically change the threshold value based on performance and load of each collection node.

#### Algorithm 1 Node load decision process

```plaintext
1: while input data is exist  
2:   threshold = input data / throughput per cycle 
3:   if channel_amount_used > 0.6 
4:     if threshold > 1 
5:       node.stat = overloaded 
6:     else 
7:       node.stat = normal 
8:   else if channel_amount_used < 0.4 
9:     if threshold < 0.5 
10:    node.stat = under-loaded 
11:    else 
12:    node.stat = normal 
13:    else 
14:    node.stat = normal 
```

**Fig. 2.** Algorithm of node load decision process

### 4 Conclusion

In this paper, we identify existing Flume system has a problem that its load balancing method completely depends on the threshold value the current user has specified for system configuration before executing it. To address this problem, we present a new load balancing method to improve the performance of Apache Flume data collector. The proposed method has
solved the problem by automatically changing the threshold value based on performance and load of each node. Therefore, the method can select the most suitable collection node based on both the increasing rate of incoming log information in the queue of each collector agent and its occupancy rate at the request of the overloaded or under-loaded collection node.

Acknowledgments. This work was supported by the Gyeonggi Regional Research Center (GRRC) and Contents Convergence Software (CCS) research center in Korea (Project No.: GRRC Kyonggi 2014-B04).

References