GOFS: A Model for Solving Overload on Distributed Stream Calculating System

Kailin Tang

Department of Computer, Guangdong University of Technology, Guangzhou 510006, China.

tkl449@126.com

Abstract. Nowadays, distributed Stream calculating is a leading technology for analyzing and managing massive streaming data. When distributed Stream calculating system working, because of data volume and distribution in the input streams often change over time, the system may be overloaded. While overload can cause the system to crash. This paper presents GOFS, a model for solving overload on distributed stream calculating system. GOFS is based on dynamic resource scheduling and load shedding. Extensive experiments are conducted to evaluate the performance and effectiveness of GOFS.

Key words: GOFS; calculating system; overloaded; dynamic resource.

INTRODUCTION

Nowadays, real-time data stream management and analysis is very valuable in many fields. Such as, the processing of massive order data of e-commerce, video monitoring and financial risk management, etc. Being strong capabilities in dealing with streaming data, distributed stream calculating system (DSCSs), such as Storm [1], Heron [2] and Flink[3], have become a popular choice for performing complex processing over data in real time. A most serious challenge of DSCSs is that streams usually fluctuate, i.e., the amount and distribution of data change over time. For example, a sales promotion often leads to a sudden surge of order for this good. Even when the data distribution remains stable, the amount of computations required can still vary from time to time. Unfortunately, the system's computational resources are limited. Therefore, the system may be overloaded, due to data fluidity. Overloading leads to longer data processing time for each input tuple, or even worse crashing the system. The conventional back-pressure [4], which pushes the unbounded data accumulation and waiting back to the source, is no longer applicable, because of the unlimited data accumulation and waiting may cause the system fail and crash. When DSCSs faced overloading, in addition to back-pressure, the load shedding [5,6], which aims at dropping tuples at certain points along the topology of stream processing application to reduce load, is also a good emergency mechanism. Attention, it's just an emergency mechanism. If you want to keep the system stable for a long time, you must use the elastic resource scheduling mechanism [7,8] to adjust the system's computing resources. Only through resource adjustment to provide sufficient computational resources can be deal with overloaded. However, load shedding mechanism inevitably affects the effective workload processed by the system, which interferences with resource scheduling. GOFS is designed for deploying resource scheduling and load shedding in a collaborative manner. By utilizing machine learning technologies to estimate the real workloads, which are reported to the resource scheduling framework, when the load shedding is enabled, GOFS is able to achieve stable of the system with fluctuating data.

RELATED WORK

The overload management problem has been studied in a lot of distributed streaming data calculating system. In order to guarantee that that each input must be completely processed, all these system, including Storm [1], Heron [2] and Flank [3], etc., use back-pressure as a emergency mechanism. Besides, the overload management problem has also been studied in the context of push-based data dissemination systems. For example, the Salamander pub-sub system supports application-level QoS policies by allowing clients to plug in their data stream manipulation modules at any point in the data dissemination tree [9]. These modules can prioritize, interleave, or discard certain data objects to adapt to changing workload and network conditions. Another overload management problem research is data stream processing system by Tatbul et al., which selectively shed streaming data among a network of data repositories to preserve user-defined latency requirements. All methodology above cannot keep the system stable for a long time, back-pressure cause system crash (due to unbounded data accumulation and waiting) and the load shedding cause low result accuracy (due to input data is not completely processed). For another, dynamic resource scheduling can be scheduling the resource to keep stable state of stream system, but its response time is too long [7,8].

DESIGN OF GOFS

To solve the deficiencies of existing methods. This paper presents GOFS, a novel collaborative model for load shedding and dynamic resource scheduling on distributed stream processing system. GOFS does not consider the implementation of resource scheduling and load shedding but focuses on designing a model to make load shedding work in coordination with dynamic resource scheduling and can effectively deal with overload situations.

Parameter Definitions and GOFS Model.

Similar to [7], the dynamic resource scheduler of GOFS focus on operator level instead of physical resource of an application, meaning that physical resource, such as CPU core, memory and disk, assigned to the operators are identical and load balancing among processors of the same operator is handled properly. In addition, we assume that the input cache for each of the operator in distributed streaming application is a first in first out queue that can implement any load shedding algorithm, including random shedding, semantic shedding and window base shedding.

Parameter Definitions

The computational framework of Distributed stream processing application is based on a directed acyclic graph (DAG), denoted as D= (V, E). And let N=|V| be the number of vertices in D. Each vertex $v_i \in V\ (1 \le i \le N)$ corresponds to a logical operator in application. each edge in D, $(v_i, v_j) \in E\ (1 \le i, j \le N)$ is represent a stream that output from v_i feeds to v_j as input. Besides, let ρ_{ij} denote the ratio of output stream from v_i that is fed to its downstream operator v_j. For each operator v_i, denotes λ_i as input arrival rate, μ_i as calcula-ting rate, ω_i as a thread of operator and $s_i()$ as the selectivity function. For example, $s_i(x) = 2x$ meaning that arrival rate of output stream is double of input stream. Finally, let δ_i denotes shedding ratio, which means that dropping probability on the operator v_i. In particular, δ_0 represents the dropping probability on the external data sources and λ_0 represents the arrival rate of external data sources. The arrival rate of external data sources is invariant, whether or not load shedding exists.

GOFS Model

Due to load shedding changes the workload, the resource scheduler cannot monitor the exact workload of system. In GOFS, by utilizing machine learning technologies, such as regression, to estimate the real workloads, which are reported to the resource scheduling framework. Therefore, the core of GOFS is to use liner regression to predict real workload. First, using a topological sort algorithm on the D, and base on topology of application, selectivity function of each operator and the external data sources workload to estimate the real workload of each operator. Therefore, the effective input arrival rate of operator becomes:

$$\lambda_i = (1-\delta_i)(\lambda_i \rho_{0i} + \sum_{j=1}^{N} \lambda_j s_j(\lambda_j)\rho_{ji}), i, j = 1,2,\ldots,N \quad (1)$$

Besides, Algorithm 1 shows the outline of the heuristic algorithm for solving the $s_i()$ in formula 1. The $s_i()$ can be obtained by regression fitting through the instantaneous input and output of operator ν_i.

```
Algorithm 1:
Input: $\lambda_{(i,in)}{}^q$, $\lambda_{(i,out)}{}^q$
Output: $s_i()$
for all q← 1,..., t do
pair[q] ← { $\lambda_{(i,in)}{}^q$, $\lambda_{(i,out)}{}^q$ }
end for
$s_i() = regression\ (pair\,[1 \ldots t])$
Return $s_i()$
```

PERFORMANCE EVALUATION

We have implemented GOFS and integrated it with Storm [1]. All experiments were run on a cluster of 10 servers running Ubuntu Linux 16.04LTS, each equipped with an Intel quad-core 2.3GHz CPU and 8GB available main memory. One machine is reserved as the master node, running Storm`s Nimbus; each of the remaining 9 machines runs up to 4 executor nodes. And we use the outlier detection application [10] for test. The outlier detection application uses the KDD Cup'99 dataset, which aims to detect network intrusions. The dataset contains over 5 million records and 42 attributes, where each record corresponds to a TCP/IP connection to the World Cup'98 website. This application consists of 3 operators, a Projection operator that projects the record onto random 1D spaces, a Detector operator that detects outliers, and an Updater that updates the result to the users as they arrive and expire. In the experiment, we made the system overloading and compared the effects of GOFS and conventional distributed stream system.

Like shown in Fig.1. Due to overload, the latency of system is very high, and the black line shows that GOFS increases the load shedding ratio and provisions more resource. By load shedding to avoiding the crash of the system and make to right decision to provisioning resource at the 3th minute. When GOFS modifies the resource configuration, the tuple latency is reduced to below the latency constraints. This indicates that DRS+ effectively maintains tuple latency to satisfy the real-time constraint. On the contrary, when there is no GOFS, the scheduler will make the wrong resource scheduling decision because it detects the distorted runtime state data, and finally cause the crash of system at 7th minute.

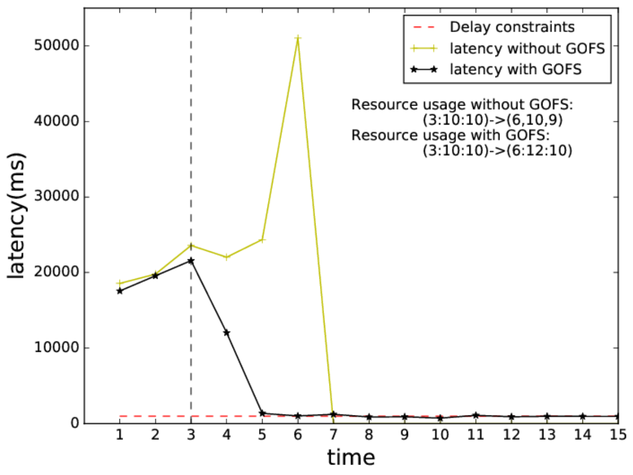

FIGURE 1. The contrast between the delay of GOFS and the delay of conventional Storm.

The experiment results show that GOFS can make resource allocation more reasonable, thus reducing latency of system and ensure the system stable.

CONCLUSION

This paper presents GOFS, it ensures stable and efficient running of the distributed stream calculating system and shows good performance in the experiment. Regarding future work directions, we plan to research and improve the state migration in GOFS.

ACKNOWLEDGMENTS

I would like to show my gratitude to all teacher who helped me to finish this research project.

The author's information is as follows. kailin Tang, postgraduate, the main research direction is algebraic theory, distributed computation and so on.

REFERENCES

1. Ankit Toshniwal, Siddarth Taneja, Amit Shukla, Karthik Ramasamy, Jignesh M. Patel, Sanjeev Kulkarni, Jason Jackson, Krishna Gade, Maosong Fu, Jake Donham, Nikunj Bhagat, Sailesh Mittal and Dmitriy Ryaboy, “Storm@ twitter” in Proceedings of the 2014 ACM SIGMOD international conference on Management of data (Snowbird, Utah, USA). ACM, 2014: 147-156.
2. Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M. Patel, Karthik Ramasamy and Siddarth Taneja, “Twitter heron: Stream processing at scale” in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (Melbourne, Victoria, Australia). ACM, 2015: 239-250.
3. Carbone, P., Katsifodimos A., Ewen S., Markl V., Haridi S. and Tzoumas, K., Bulletin of the IEEE Computer Society Technical Committee on Data Engineering. 36(4), (2015).
4. L. Georgiadis, M. J. Neely, and L. Tassiulas, Foundations and Trends in Networking, vol. 1, no. 1, pp. 1-149, (2006).
5. Rivetti, N, Busnel, Y, & Querzoni, L. Load-aware shedding in stream processing systems. In Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems (pp. 61-68). ACM. (2016, June).
6. E. N. Tatbul, Load shedding techniques for data stream management systems. PhD Thesis, Brown University, Rhode Island, 2007.
7. T. Z. Fu, J. Ding, R. T. Ma, M. Winslett, Y. Yang, and Z. Zhang, IEEE/ACM Transactions on Networking, vol. 25, no. 6, pp. 3338–3352, (2017).
8. Madsen, Kasper Grud Skat, and Yongluan Zhou, “Dynamic resource management in a massively parallel stream processing engine.”, in Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. (Melbourne, Australia). ACM, 2015.
9. G. R. Malan, F. Jahanian, and S. Subramanian. Salamander: A Push-based Distribution Substrate for Internet Applications. In USENIX USITS Symposium, Monterey, CA, December 1997.
10. Tian Tan, Richard T.B. Ma, Marianne Winslett, Yin Yang, Yong Yu, Zhenjie Zhang, “Resa: realtime elastic streaming analytics in the cloud” (New York, New York, USA), in Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. ACM, 2013: 1287-1288.