Advances in Intelligent Systems Research, volume 147

International Conference on Network, Communication, Computer Engineering (NCCE 2018)

Physical Computing Resource Scheduling on Distributed Stream Processing System

Kailin Tang

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

tkl449@126.com

Abstract. Recently, distributed stream computing has emerged as a popular model for real-time data stream analytics. This paper focuses on computing resources scheduling of distributed stream processing. To adapt to the fluctuation of data stream, distributed stream processing system must elastically provision the computing resources based on the observed workload. However, there is no system that satisfies all these requirements. Motivated by this, this paper presents a physical resource scheduling algorithm for elastic scheduling computing resources of stream processing task. Base on existing task tier elastic resource scheduling and system-tier resource scheduling, this paper proposes a collaborative algorithm. According to the fluctuant workload, task-tier dynamic resource scheduler config computing resource of stream processing task. Meanwhile, based on the scheduling decision of task-tier scheduler, the system-tier resource scheduler is assign physical computing resources to stream processing tasks.

Key words: real-time data; workload; system-tier; stream processing.; elastic scheduling.

INTRODUCTION

Nowadays, more and more enterprises and individuals are beginning to use distributed streaming data processing systems to analyze and manage their massive real-time data. A common challenge faced by distributed stream processing systems(DSPSs) is that streams usually fluctuate. The workload may greatly exceed the system processing capability, or far less than the system processing capability. Usually, distributed stream processing systems, such as Storm [1], Heron [2], Flink[3], involves a task-based resource scheduler that dynamically provisions resources to operators to handle stream fluctuations. To avoid overloading, the scheduler modifies the computing resources of the data stream processing task in real time by monitoring the workload. The physical computing resources available to the DSPSs are constant. Therefore, dynamic resource scheduling can only be running on limited physical resource [4]. This is not a complete resource scheduling, and in many cases the need for resources varies greatly. Limited physical resources may result in resource scheduling failures and eventually result in a system degradation or even crash. Currently, In the field of distributed stream data processing, there are some system-tier resource scheduling models that can be used for scheduling physical resources, such as Storm on yarn [5], Flink on yarn [6] and Storm on Mesos [7] etc. The Yarn [8] and Mesos [9] were originally a resource scheduling framework for batch systems, both of which can only allocate resources statically instead of dynamically. Therefore, Storm on yarn or other similar system cannot scheduling physical resources based on monitor workload. Motivated by creating a real sense of dynamic scheduling for distributed stream processing, this paper presents a physical computing resources scheduling algorithm, which using task-tier dynamic resource scheduler to config computing resource of stream processing task, which executed by the system-tier resource scheduler.

RELATED WORK

Resource scheduling is a hot research topic in cloud computing and distributed computing. In area of highly-parallel batch data processing, resource scheduling has been a central problem. and a lot of schedulers have been developed and used in production, e.g., Fair Scheduler [10], Capacity Scheduler [11]. In addition, there are some complex distributed scheduling frameworks that contain a variety of scheduling algorithms, such as Yarn [8] and Mesos [9]. The above model and framework are called the system-tier scheduler. Since the amount of data for batch tasks is constant, batch tasks resource scheduling is much easier than streaming tasks. And this system-tier scheduler cannot satisfy all requirement of distributed stream processing. Meanwhile, according to the research of [4] and [12], dynamic resource scheduling on distributed stream processing system is based on virtual computing resource, such as a processor or a thread, and this scheduler can only be running on limited physical resource. All in all, there is no complete resource scheduling scheme in the distributed stream processing system.

PHYSICAL COMPUTING RESOURCE SCHEDULING

To create a complete dynamic physical resource scheduling in the distributed stream processing system, this paper presents a physical computing resource scheduling algorithm. This algorithm designed for deploying task-tier dynamic resource scheduling and system-tier scheduler in a collaborative manner. First, we introduced the distributed stream processing model and then described the algorithm.

Distributed Stream Processing Model

The distributed stream processing model is based on a directed acyclic graph (DAG), and each vertex corresponds to a logical operator in the task, each directed edge denotes a data stream. We assume that there is an operator in the task. We define ri as the number of virtual computing resource for ith operator. Besides, we define M which is the aggregation of all the task status, including resources status, information that monitoring by task-tier resource scheduler.

Physical Computing Resource Scheduling Algorithm Description

In reality, there are many different task-based resource scheduling models and a lot of physical resource-based system-tier scheduler. For ensuring the algorithm more general, this paper do not consider the implementation of system-tier scheduling algorithm and task-tier scheduling algorithm. The specific choice of the scheduling algorithm is domain-specific and, thus, is out of the scope of this work. Our only requirement for the task-tier scheduler is that the output of task-tier scheduler must a vector that like following shown:

$$\vec{r} = (r_1, r_2, ..., r_n) \tag{1}$$

As shown in formula 1, this vector is to characterize a resources assignment, where r_i of this vector represents the number of virtual computing resource unit for ith operator of the stream task. Let $p = (c, m, d)$ denotes a physical computing resource allocation. The c denotes a unit of CPU, the m denotes a unit of memory and the d denotes a unit of disk. The network costs are not explicitly considered. In addition, let a pair $(\vec{r}, p)$ denotes the resource configuration made by system-tier scheduler, $\vec{r}$ is the output of task-tier scheduler and the p is the physical resource assignment. The "unit" above can be any module and can configure by users.

Algorithm 1 shows the outline of the collaborative physical resource scheduling algorithm.

Algorithm 1:
Input: M
Output: $(\vec{r}, p)$
$\vec{r}_{new} \leftarrow$ Task-tierScheduler(M)
if $\vec{r}_{new} \neq \vec{r}_{old}$ then
Derive $|\vec{r}_{new}|$ according to $\vec{r}_{new}$

physical_resource_request ← CoordinateScheduler($|\vec{r}_{new}|$,M)
p ←System-tierScheduler(physical_resource_request)
$(\vec{r}, p) \leftarrow \vec{r}_{new}$, p
end if
Return $(\vec{r}, p)$

As shown on Algorithm 1. CoordinateScheduler is the core of the collaborative algorithm. By converting the task layer output to the system layer input, CoordinateScheduler makes the physical scheduling possible. Besides, the physical_resource_request in algorithm 1 is depends on the implementation of the system-tier scheduler, this is out of the scope of this work and we are not explicitly considered.

EXPERIMENT AND EXPERIMENTAL RESULT ANALYSIS

We have implemented physical resource scheduling model and integrated it with Storm on yarn [5]. and in our implementation, the task-tier resource scheduler is DRS [4], the system-tier resource scheduler is Yarn. The experiments are run on a cluster with 10 desktop PCs connected to the same router. Each machine has an Intel quad-core 2.3GHz CPU and 8GB available main memory. We are configuring that Storm`s Nimbus running on a PC and each supervisor running on a PC respectively. Besides, each of the machines running supervisor runs up to 4 executors. For performance evaluation, we are submitting a data stream frequent pattern detection task and the dataset used for the frequent pattern detection task, which contains about 20 million tweets [13]. And this task consists of 3 operators, a Generator operator that split the twitter to a set of words, a Detector operator that detects maximal frequent item in real-time, and an Reporter that report the result to the users.

Fig.1 shows the resource allocation changes, system starting with 36 unit of physical resource, and only 20 unit in use. After resource scheduling, the total number of system resources becomes 17, equal to the resources used. 17 is the $|\vec{r}_{new}|$.

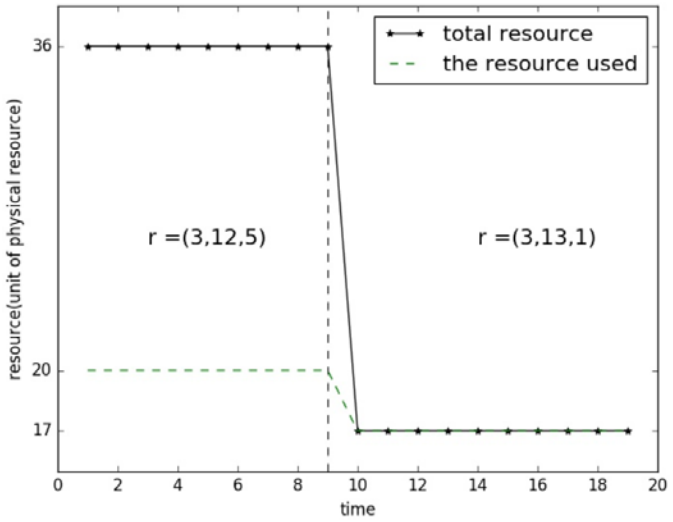

FIGURE 1. Resource changes of distributed stream processing system.

The evaluation results indicate that physical resource scheduler can allocate physical resource in distributed stream processing system more reasonable and high-utilization.

CONCLUSION

In this work, we have studied elastic virtual resource in distributed stream processing area and studied physical resource scheduling model. We have presents a physical resource scheduling algorithm for elastic scheduling physical resource on distributed stream processing system. And the physical resource scheduling algorithm performs

well in evaluation. However, there are many theory worth to more time for an in-depth study of it, such as how to reduce the communication cost between task-tier dynamic resource scheduler and system-tier resource scheduler.

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. 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).
5. Yahoo Inc. Storm on YARN. https://github.com/yahoo/storm-yarn.
6. Apache Flink on YARN. https://ci.apache.org/projects/flink/flink -docs-release-1.3/setup/yarn _setup. html#flink-yarn-session.
7. Apache Storm on Mesos. https://github.com/mesos/storm.
8. Vinod Kumar Vavilapalli, Arun C. Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed and Eric Baldeschwieler, "Apache Hadoop yarn: Yet another resource negotiator," in Proceedings of the 4th annual Symposium on Cloud Computing (Santa Clara, California). ACM, 2013, p. 5.
9. B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center.", NSDI. 2011, 11(2011): 22-22.
10. Fair scheduler, http://hadoop.apache.org/docs/r1.2.1/fair scheduler.html.
11. Capacity scheduler, http://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html.
12. 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.
13. Twitter dataset. http://dx.doi.org/10.18170/DVN/EF9YLX.