ATLANTIS PRESS

Advances in Engineering Research, volume 114

International Conference on Advances in Materials, Machinery, Electrical Engineering (AMMEE 2017)

Performance-sensitive components exploration in Spark Streaming

Ying Hou [a], Yi Liang [b] and Chao Su [c]

School of Beijing University of Technology, Beijing 100124, China;

[a]houying109@ emails.bjut.edu.cn, [b] yliang@bjut.edu.cn, [c]13810469621@163.com

Keywords: big data, spark streaming, performance-sensitive components.

Abstract. Streaming data processing has become a hot topic in the big data research. To ensure the timeliness of data processing, it is important to explore the performance-sensitive components in the streaming data processing platform, which can contribute to the more efficient performance optimization. In this paper, we describe the data processing model in the Spark Streaming, the process can be divided into multiple phases. We propose a simple yet useful method to explore performance-sensitive component components among these phases. Experimental results show that the proposed method is suitable for a wide range of workloads. At last, we demonstrate a detail example of the application of this method on the typical Spark Streaming workload Word count and prove its practicability.

1. Introduction

Spark Streaming is the cutting-edge system for the batch-based streaming data processing. It meets the wide demands in real-time sensor data processing [1,2] and social network stream data analysis [3,4]. Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams[5].

Spark streaming is a complex system consisting of multiple data processing phases, including data reception, storage, processing, and so on. For each phase, there is a component that dominates the data processing performance, which is called the performance-sensitive component. The performance-sensitive components can be various with different hardware/software infrastructures. It is obvious that exploring the performance-sensitive component can guide the performance optimization of Spark Streaming system more efficiently. However, to our best knowledge, it is still a blank area in the Spark Streaming research.

In this paper, we firstly describe the abstract multiple-phase model of data processing of Spark Streaming, and then propose a method that identify performance-sensitive component components with the sojourn time of each phase. We show through extensive experiments that the proposed method is suitable for a wide range of workloads.

In the rest of the paper, Section 2 introduces the principle of Spark Streaming, and present the data processing model. Section 3 describes the exploration method of performance-sensitive components in detail. Section 4 gives the experimental results and Section 5 finalizes the paper with a brief conclusion.

2. Behavior Analysis Of Spark Streaming

2.1 Principle of Spark Streaming

In Spark streaming system, the streaming data is divided into mini-batches and stored in Spark's memory as RDDs (Resilient Distributed Datasets), and then periodically processed by Spark engine with the MapReduce-styled batch computation [6].

There are two important configuration parameters in Spark Streaming, one is block interval, the other is batch interval. For each block interval, the incoming data items in the DStream are generated as a data block and stored in Spark. For each batch interval, the new-generated data blocks are abstracted as an RDD and submitted to be processd as a Spark job.

2.2 Phase Division of Spark Streaming

We divide the data processing procedure into two parts,the data acquisition and the data processing,. Each part consists multiple phases as in Figure 1.

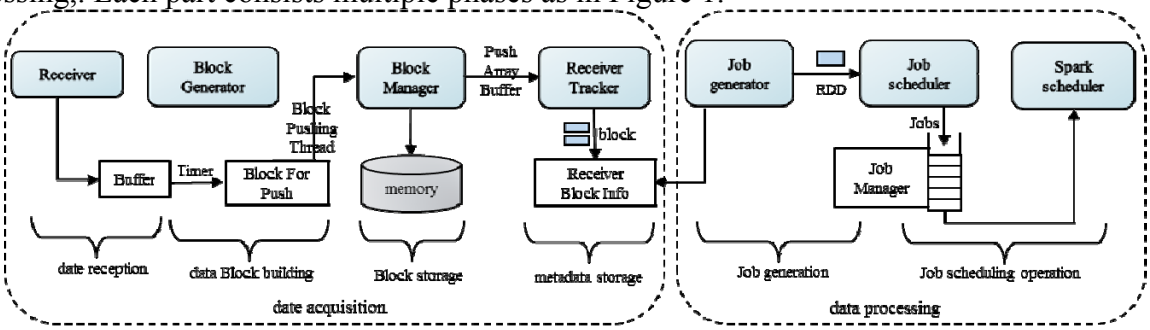

Fig. 1 Phase division of Spark Streaming

The processing phases are described as follows:

(1) data reception. Reveiver component ingests stream data from data sources (such as Kafka), and store it in memory buffer.

(2) data block building. Timer of Block Generator component periodically pack the data in buffer up a data block after a block interval by user-defined, and puts the data block into Block For Push queue. The block interval is 200ms acquiescently.

(3) Data block storage. Block Pushing Thread thread continually passes the blocks in the Block For Push queue to the Block Manager component, which store data block int memory or disk.

(4) Metadata storage. Push Array Buffer method passes the metadata information (such as blockId) of the data block stored by the Block Manager to the Receiver Tracker component, and the Receiver Tracker component puts blockId into the corresponding stream queue and store blockId in the Receiver BlockInfo queue.

(5) Job generation. Job Generator component periodically extract all blocks in Receiver BlockInfo queue to generate the corresponding RDD, and generate job according to the RDD dependency. The job is submitted to Job Queue.

(6) Job scheduling operation. Job Scheduler schedules job form Job Queue, and submits to Spark's Scheduler. The job is split into a large number of tasks and is processed on cluster's Executor component.

3. Performance-sensitive Components Exploration

3.1 The Significances of Exploring Performance-Sensitive Components

According to the mentioned in section 2, data processing in Spark Streaming is composed of multiple phases. And a phase corresponds to a core component. The core components of these phases are Receiver, Block Generator, Block Manager, Receiver Tracker, Job Generator and Executor respectively.

In the process of data processing, the data sequentially goes through multiple components, and the sojourn time of each component is different. Selecting performance-sensitive components is the premise of optimizing the performance of system, which can maximally improve system performance efficiency.

3.2 Method of Exploring Performance-Sensitive Component

The core idea of exploring performance-sensitive component is to collect sojourn time of each phase, which is used to compute the percentage of sojourn time and standard deviation of each phase. The percentage of sojourn time can evaluate the criticality of the component, and standard deviation can reflect the randomness of sojourn time of component.

The following variables and definitions are used for the method.

$C=\{c_i \mid 1 \le i \le 6\}$: the set of core components that described in part A.

$D=\{d_j \mid 1 \le i \le \mathrm{n}\}$: the set of test data item.

$\lambda=\{\lambda_m \mid 1 \le \mathrm{m} \le \mathrm{p}\}$: the set of data arrival rate.

P: the percentage of sojourn time.

σ: standard deviation.

C_K: the set of performance-sensitive component components.

C_R: the set of random components, namely the sojourn time of component is random.

C_K: the set of constant components, namely the sojourn time of component approximately is a fixed value.

T_{endi}: the time that data d_j leave component c_i.

T_{starti}: the time that data d_j come component c_i.

T_m: total sojourn time of data d_j under data arrival rate λ_m, namely the sum of the average sojourn time of each component.

P_a: the threshold value of the percentage of sojourn time.

σ_a: the threshold value of standard deviation.

Baseed on the above variables and defination, the method is described as follows.

Step 1: For data arrival rate $\lambda_m \in \lambda$, and data item $d_j \in D$, the sojourn time of component c_i.is T_{ijm}, it is calculated according to

$$T_{ijm} = T_{endi} - T_{starti} \tag{1}$$

Step 2: The average sojourn time of component c_i. under data arrival rate λ_m is T_{im}, it is calculated according to

$$T_{im} = \frac{\sum_{j=1}^{n} T_{ijm}}{n} \tag{2}$$

Step 3: T_m is total sojourn time of data item d_j under data arrival rate λ_m, it is calculated according to

$$T_m = \sum_{i=1}^{6} T_{im} \tag{3}$$

Step 4: For $\lambda_m \in \lambda$, the percentage of average sojourn time of component c_i is P_{im}, and the standard deviation of component c_i is σ_{im}, they are calculated according to

$$P_{im} = \frac{T_{im}}{T_m} \tag{4}$$

$$\sigma_{im} = \sqrt{\frac{\sum_{j=1}^{n} (T_{ijm} - T_{im})^2}{n}} \tag{5}$$

Step 5: The percentage of average sojourn time of component c_i under different data arrival rate is P_i, and the average standard deviation of component c_i under different data arrival rate is σ_i, they are calculated according to

$$P_i = \frac{\sum_{m=1}^{p} P_{im}}{p} \tag{6}$$

$$\sigma_i = \frac{\sum_{m=1}^{p} \sigma_{im}}{p} \tag{7}$$

Step 6: For each $c_i \in C$, if $P_i \geq P_a$, then $C_K = C_K \cup \{ c_i \}$.

Step 7: For each $c_i \in C_K$, if $\sigma_i \geq \sigma_a$, then $C_R = C_R \cup \{ c_i \}$, else $C_C = C_C \cup \{ c_i \}$.

4. Experiment And Analysis

4.1 Experimental Setup

We conduct the experiments in a cluster of 7 machines, each equipped with Intel Xeon E5-2660 processors and 16GB physical memory. Each processor contains six processing cores. The operating system for the cluster is centos 6.5 and all machines connected through 1 Gigabit Ethernet. In our cluster, computation cluster contains 5 nodes, one functioning as master and others as slaves. One is data generator node that generate data and push it to Apache Kafka, the last one node for deploying

Apache Kafka and Zookeeper, Kafka send data to computation cluster, Zookeeper provides services for Kafka and Apache Spark.

We use wordcount, a typical streaming workload, under different data arrival rates to test. We use Hibench benchmark as dataset generator. Table 1 shows the group of data arrival rate.

4.2 Experiment Results

We run wordcount application under different data arrival rates on Spark Streaming.

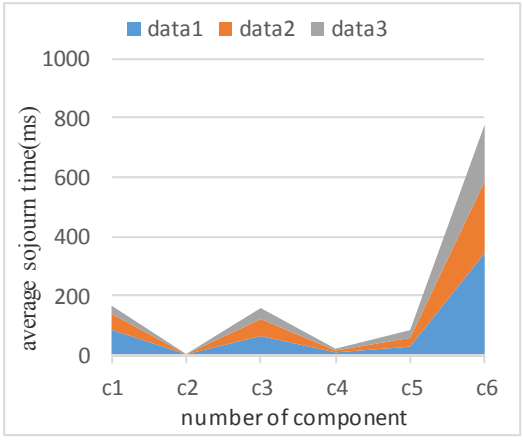

Fig. 1 The sojourn time of component under different data arrival rate

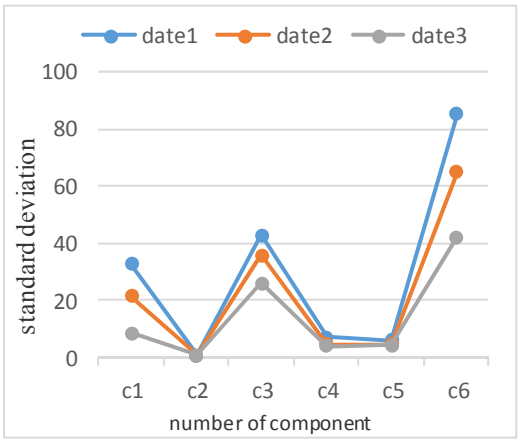

Fig. 2 The standard deviation of component under different data arrival rate

Fig. 1 presents the sojourn time of each component under different data arrival rates. In the process of data processing, the sojourn time of component c_1, c_3, c_6 is longer than component c_2, c_4, , c_6. As the changes of the data arrival rate from data1 to data3, the sojourn time of the same component is decreasing.

In this paper, we use the percentage of sojourn time and standard deviation to explore performance-sensitive components. Fig. 2 show the standard deviation of each component. Table 2 presents the percentage of sojourn time of all components for each data arrival rate. The percentage of sojourn time of c_6 is the biggest.

Table 1. The Percentage of Sojourn Time under Different Data Arrival Rate

Number of component	Data1	Data2	Data3
c_1	15.92%	13.96%	9.51%
c_2	0.18%	0.21%	0.21%
c_3	12.00%	14.65%	12.82%
c_4	1.46%	1.68%	2.02%
c_5	5.12%	7.52%	9.17%
c_6	65.32%	61.98%	66.27%

Table 2. The Percentage of Sojourn Time under Different Data Arrival Rate

Number of component	the percentage of average sojourn time	average standard deviation
c_1	13.13%	18.57
c_2	0.20%	0.84
c_3	13.16%	28.42
c_4	1.72 %	4.89
c_5	7.27%	5.24
c_6	64.52%	48.38

The experimental results of the percentage of average sojourn time and average standard deviation of component are shown in Table 3. We configured our parameter to use P_a =10% and σ_a = 10. The set of performance-sensitive component components C_K.= { c_1, c_3, c_6}={Receiver, Block Manager, Executor}, the set of random components C_R.= { c_1, c_3, c_6}={Receiver, Block Manager, Executor}, the set of random components C_C.= ∅ .

5. Conclusion

In this paper, we have proposed a simple yet useful method to explore the performance-sensitive components for Spark Streaming. We also show the practicability of this method with the typical Spark Streaming workload.

References

[1]. Yang K L, Ling W, Ryu K H.A System Architecture for Monitoring Sensor Data Stream[C]// International Conference on Computer and Information Technology, 2007: 1026-1031.

[2]. Martinez-Julia P, Torroglosa Garcia E, Ortiz Murillo J, et al.Evaluating Video Streaming in Network Architectures for the Internet of Things[C]//International Conference on Innovative Mobile & Internet Services in Ubiquitous Computing, 2013:411-415.

[3]. Ediger D, Riedy J, Bader D A, et al.Tracking Structure of Streaming Social Networks[C]//IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, 2011:1691-1699.

[4]. Borthakur D, Gray J, Sarma J S, et al.Apache hadoop goes realtime at Facebook[C]//ACM SIGMOD International Conference on Management of Data, 2011:1071-1080.

[5]. Spark Streaming. http://spark.apache.org/docs/latest/streaming-programming-guide.html.

[6]. Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing[C]// Usenix Conference on Networked Systems Design and Implementation. USENIX Association, 2012:2-2