Water-Wheel: Real-Time Storage with High Throughput and Scalability for Big Data Streams

Yanqi Lv, Ruicheng Liu, Peiquan Jin
[1]*School of Computer Science and Technology, University of Science and Technology of China, Hefei, China*
[2]*Key Lab. of Electromagnetic Space Information, Chinese Academy of Sciences, Hefei, China*
jpq@ustc.edu.com

***Abstract*—In this demonstration, we present a real-time storage system called Water-Wheel for big data streams. Big data streams involve rapid and continuous data flows, which will overwhelm the bandwidth capacity of conventional disk storage systems. The new Water-Wheel system proposes a new storage model that employs multiple nodes (servers) to accept data flows using a rotation way, like a traditional water wheel. When a node becomes full, it is transformed into a data-dumpling state that will flush the data in the node to durable storage, after which the node will become empty again to accept new data flows. We demonstrate that the Water-Wheel system can offer higher and more scalable storage bandwidth than existing big data storage systems that use a master-slave architecture to handle big data streams. After an introduction to the architecture and key designs of Water-Wheel, we present a case study to demonstrate Water-Wheel.**

***Keywords*—Big data, Real-time storage, Throughput, Scalabity**

I. Introduction

The rapid development of the Internet of Things (IoT) [1] leads to big data streams in many applications. For example, there will be over 100 millions data inputs every second in a smart-grid monitoring application. Such highly-arriving big data makes it difficult to persist data in the data center using existing approaches [2], [3].

In this paper, inspired by the ancient water wheel in China, which can transport water from one spot to another through some rotating buckets, we propose a new real-time storage system called *Water-Wheel* for big data streams. We devise a rotation storage model [4] that can distribute data flows among multiple data nodes. This approach can make full use of each node in the data center and maximize the write throughput of the data center. Briefly, the Water-Wheel system has the following unique features:

(1) **High Throughput**. It employs a rotation storage model to deal with the conflict between highly-arriving data stream and low write throughput of the underlying storage. According to the rotation storage model, we set the memory of each node as a data bucket and all data buckets are rotated from the state of idle waiting to data filling, write waiting, and data dumping. With this mechanism, we can use a few data buckets to meet the high-throughput need required by big data streams.

(2) **High Scalability.** Water-Wheel is implemented based on the share-nothing architecture. New data nodes can be easily appended to the system and its storage capacity will be automatically added to the system. In case of increasing data flows, Water-Wheel offers high scalability by adding nodes seamlessly.

DOI reference number: 10.18293/SEKE2021-204

II. Architecture of Water-Wheel

The key design of Water-Wheel is a rotation storage model [4], which consists of a set of data buckets. A data bucket corresponds to a buffer in a data node in a cluster. All data nodes form a huge storage space that can be regarded as a ring because each data node is used with a round-robin manner.

All the buckets work according to the process indicated by arrows in Fig. 1. We can see that the Water-Wheel works in memory and the underlying storage nodes provide persistent storage service. When a bucket is filled in memory, it will be persisted to some storage node. We manage the states of all buckets and develop a state-transition scheme to make each bucket work at a right manner. As shown in Fig. 1, each bucket has one specific state at every moment. There are four states designed for buckets:

- State 1: *Idle Waiting*.
- State 2: *Data Filling*.
- State 3: *Write Waiting*.
- State 4: *Data Dumpling*.

Note that each bucket changes its state according to the given sequence, i.e., from State 1, 2, 3, and 4. Then, it will repeat the state transition from State 1 to 4. With such a mechanism, if one bucket becomes full, it will be changed into the state of *Data Dumpling*. And after we write the bucket to persistent storage, we can reuse the bucket, meaning that we can put the bucket into the waiting queue and let it wait to accept new data insertions. This is done by setting the bucket' state to *Idle Waiting*.

Water-Wheel is deployed as a middle layer between the data stream and the underlying storage node. Combined with the state transition of data buckets, we give the general workflow of Water-Wheel in Fig. 2:

(1) When the data stream arrives, Water Wheel will first save the data in the data buckets in the state of data filling. Note that the number of the data buckets in the state of data filling can be configured in advance. Generally, more data-filling buckets can accept a bigger data stream. However, it will also incur higher pressure when dumping the data from

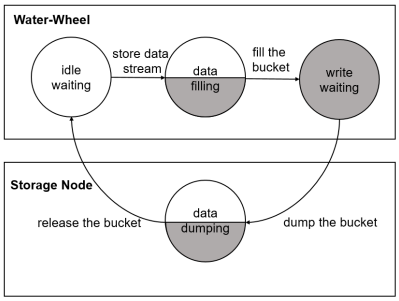

Fig. 1. The state transition of a data bucket

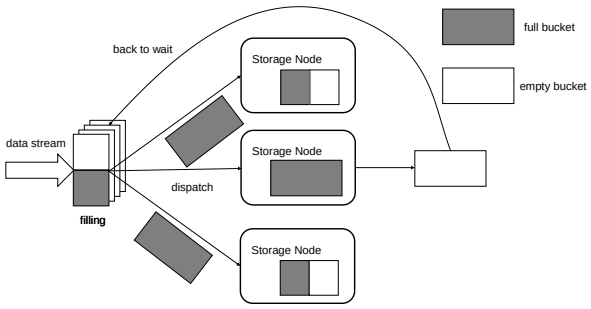

Fig. 2. The working process of Water-Wheel

memory to persistent storage. We set the number of data-filling bucket to one by default.

(2) The a data-filling bucket becomes full, we change its state from data filling to write waiting to let the bucket not accept newly-arriving data streams.

(3) all the buckets in the state of write waiting will be dispatched to some storage nodes, which will be responsible for moving the data from memory to persistent storage.

(4) When a storage node gets a bucket in the state of write waiting, it will change the state of the bucket into data dumpling. Then, the storage node starts to move the data in the write-waiting bucket to files in persistent storage, e.g., SSDs or HDDs.

(5) After a storage node has completed the data dumpling of a bucket, which means that all the data in the bucket has been persisted, it will change the bucket's state from data dumpling into idle waiting. And all idle-waiting buckets are organized into a queue. When newly-arriving data streams come to the system, we will select one ore more idle-waiting buckets from the queue to accept data streams. Meanwhile, those selected buckets will be marked as the state of data filling.

The above process will be executed repeatedly when the data stream comes to the system continuously. As all data buckets are reused to accept newly-inserted data, we can infer that the system can deliver a high write throughput. Even when the data stream becomes extremely fast, e.g., more than 10GB data written in one second, we can configure more data-filling buckets and let each bucket be equipped with large memory (e.g., using persistent memory [5], [6]), so that the system can absorb more than 10GB data in one second.

III. Demonstration

We implemented Water-Wheel on top of MongoDB [3]. We designed a graphical user interface for Water-Wheel, as shown in Fig. 3. User can monitor the real-time write throughput of the system. Users can also monitor the state of each node, such as the CPU utilization and write throughput. Water Wheel provides high scalability, meaning that we can easily add a new node into the system. Figure 4 shows the interface of adding a new node to the system. In the future, we will deploy our system to a cloud storage platform, so that it can support experiments running on hundreds of nodes.

Fig. 3. The main interface of Water-Wheel

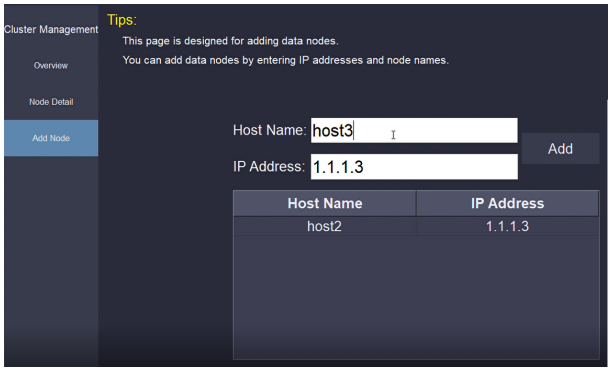

Fig. 4. Adding a new node to the system

Acknowledgements

This work was supported by the National Science Foundation of China (62072419). Peiquan Jin is the corresponding author.

References

[1] D. Jung and et al., "Vibration analysis for iot enabled predictive maintenance," in *ICDE*, 2017, pp. 1271–1282.
[2] X. Hao and et al., "Efficient storage of multi-sensor object-tracking data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2881–2894, 2016.
[3] MongoDB, 2021. [Online]. Available: https://www.mongodb.com/
[4] Y. Lv and P. Jin, "Rotaryds: Fast storage for massive data streams via a rotation storage model," in *CIKM*, 2020, pp. 3305–3308.
[5] Z. Wu, P. Jin, C. Yang, and L. Yue, "APP-LRU: A new page replacement method for pcm/dram-based hybrid memory systems," in *Proc. of NPC*, 2014, pp. 84–95.
[6] K. Chen, P. Jin, and L. Yue, "A novel page replacement algorithm for the hybrid memory architecture involving PCM and DRAM," in *Proc. of NPC*, 2014.