# The Tentative Research of Hydrological IoT Data Processing System Based on Apache Flink

Feng Ye[1,2](✉), Peng Zhang[3], Cheng Hu[1], Songjie Zhu[1], and Ling Li[1]

[1] College of Computer and Information, Hohai University, Nanjing, China
yefeng1022@hhu.edu.cn

[2] Postdoctoral Centre, Nanjing Longyuan Micro-Electronic Company, Nanjing, China

[3] Jiangsu Province Water Resources Department, Nanjing, China

**Abstract.** With the widespread application of sensor and IoT technology in the field of water conservancy informatization, the traditional application systems based on Java EE or pure NoSQL databases for hydrological data processing and analysis have been difficult to meet the new requirements for processing and analyzing large-scale hydrological IoT stream data. How to select a suitable big data processing platform and how to implement application systems for hydrological IoT stream data requires in-depth theoretical foundations, more experimental comparisons, effective design paradigm and practical implementations. This paper summarizes the research status of big data in water conservancy domain, and then proposes a hydrological IoT data processing system based on Apache Flink. We use the sensor data obtained in Chuhe river as the experimental dataset, and take the common and daily operations for hydrological data as example. The experimental results show that the processing capability of the hydrological IoT data processing system is far superior to the traditional multi-tier architecture system based on Java EE or pure NoSQL databases, and it obviously becomes an appreciable solution for water conservancy informatization.

**Keywords:** Apache Flink · Stream data · Water conservancy informatization · IoT

## 1 Introduction

With the widespread use of Internet of things (IoT) [1] and the rapid development of the mobile Internet, diverse data has been growing explosively. How to deal with large-scale IoT data has become a research hotspot [2]. Meanwhile, in the field of water conservancy informatization, the hydrological IoT data acquisition capability has also continuously improved. Because of the diversity, dynamics and large-scale of the hydrological IoT stream data, the traditional multi-tier architecture system based on Java EE or pure NoSQL databases for hydrological data processing and analysis have been difficult to meet the new requirements for processing and analyzing these hydrological IoT stream data. In this context, how to select a suitable big data processing platform and how to implement an appreciable solution for hydrological IoT stream data become a key challenge. According to existing research and the present situation of the information development in water conservancy informatization, we think it requires in-depth theoretical foundations, more experimental comparisons, effective design paradigm and practical implementations.

By comparing the mainstream big data processing platforms, we propose a novel hydrological IoT data processing system based on Apache Flink [3, 4], which is an open source framework and distributed processing engine for stateful computations over unbounded and bounded data streams. To our knowledge, though Apache Flink powers business-critical applications in many companies and enterprises around the globe, such as Alibaba.com and Ericsson, there are no cases in the field of water conservancy informatization. Using the sensor data obtained from Chuhe river as the experimental data, the proposed system is proved to be advanced in water conservancy informatization domain.

The rest of the paper is organized as follows. Section 2 describes some works related to this topic of interest. In Sect. 3, after introducing the Apache Flink, the architecture and components of the proposed hydrological IoT data processing system is described. In Sect. 4, comparing with the traditional multi-tier architecture system based on Java EE or pure NoSQL databases, we analyzes the performance of the proposed the hydrological IoT data processing system using the sensor data obtained in Chuhe river. At last, conclusion along with the direction for future research is provided in Sect. 5.

## 2 Relate Works

According to Feng [5], after long-term application practice, a large number of heterogeneous business data have been accumulated in water conservancy domain. By 2012, the hydrological data alone had exceeded 100 TB nationwide. With the development and widespread application of IoT related technologies such as remote sensing, sensor and so on, the hydrological IoT data acquisition capability has been continuously improved, and more and more hydrological data have been collected and utilized in water conservancy informatization. These hydrological IoT data often have the following characteristics [6]: (1) Multi-source: data is captured by sensors at different locations; (2) Heterogeneous stream data; (3) Thematic diversity: there are many topics such as water quality, hydrology and irrigation, and different topics require different computing patterns. (4) Presence of outliers: observations considerably higher or lower than most of the data, which infrequently but regularly occur. (5) Autocorrelation: consecutive observations tend to be strongly correlated with each other. (6) Dependence on other uncontrolled variables: values strongly co-vary with water discharge, hydraulic conductivity, sediment grain size, or some other variable. From the perspective of data type, these massive data includes both batch and stream data. From the perspective of timeliness, certain hydrological data such as flood warnings require timely and efficient processing and feedback. Moreover, because the data lineage for location, environment, weather and other related factors is often missing in the process of data acquisition, there is a wealth of spatial-temporal correlation information between data lost. Therefore, when the traditional hydrological data processing systems based on Java EE or pure NoSQL databases cannot effectively face the large-scale hydrological IoT data, it is necessary to adopt a suitable big data processing engine to improve the processing capabilities of such data [7].

Currently, there are already many typical big data processing platforms. The Map-Reduce [8] framework has become a de facto standard for big data technology and is widely used to manage large clusters. Hadoop [9] is an open source implementation of the MapReduce framework and plays an active role in the big data technology system. However, in practice, industry and academia also gradually find that the MapReduce framework and Hadoop implementation are not one-size-fits-all big data processing solutions [10]. For example, the Hadoop platform is not suitable for second - or micro-second interactive queries [11]. Therefore, Map-Reduce and Hadoop are difficult to apply to hydrological IoT data processing effectively. Apache Spark [12, 13], is an another large data parallel computing framework based on in-memory computing that can be used to build large, low-latency data analysis applications. On the speed side, Apache Spark extends the popular Map-Reduce model to efficiently support ore types of computations, including interactive queries and stream processing. On the generality side, Apache Spark is designed to cover a wide range of workloads that previously required separate distributed systems, including batch applications, iterative algorithms, interactive queries and streaming. Although the throughput has improved, the biggest problem is that Apache Spark lacks low end-to-end latency with exactly-once guarantees [14–16]. Obviously, they are unable to satisfy some high throughput and low latency processing scenarios, such as flood warning and forecasting.

Compared with mainstream big data platforms like Apache Spark and Hadoop, Apache Flink is considered to be the fourth generation and the latest generation big data processing engine. Specifically speaking, Apache Flink is a framework and distributed processing engine for stateful computations over bounded and unbounded data streams. It has been designed to run in all common cluster environments, perform computations at in-memory speed and at any scale. The core computational fabric of Apache Flink, labeled “Flink runtime” in Fig. 1, is a distributed system that accepts streaming dataflow programs and executes them in a fault-tolerant manner in one or more machines. Apache Flink also offers developer-friendly APIs that layer on top of the runtime and generate these streaming dataflow programs.

As far as I know, though Apache Flink powers business-critical applications in many companies and enterprises around the globe, such as Alibaba.com and Ericsson, there are no cases in the field of water conservancy informatization. Therefore, from the above, we can see that Apache Flink is a versatile processing framework that can handle any kind of stream, and it is necessary to study how to design and implement IoT data processing system in combination with Apache Flink in water conservancy domain.

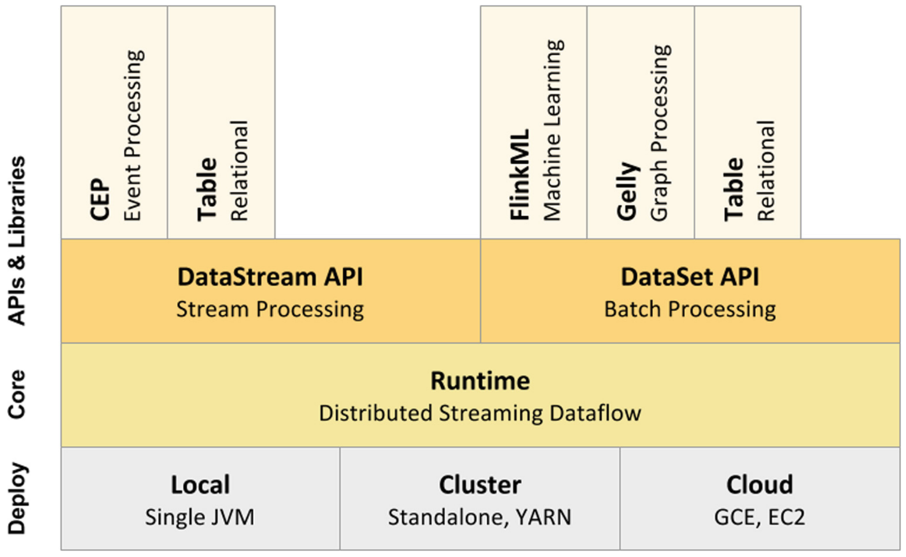

**Fig. 1.** The key components of the Apache Flink stack

## 3 The Proposed Hydrological IoT Data Processing System

The system architecture proposed is shown in Fig. 2, and there are four tiers: infrastructure layer, virtualization layer, dataset processing layer and visualization layer. The infrastructure layer provides the hardware foundation for big data processing, such as PCs, various servers and network equipment. Various resources are abstracted into different resource pools, such as data resource pool, network resource pool.

In virtualization layer, Apache CloudStack is installed, configured and deployed to construct virtual machines cluster and then used to manage the infrastructure resource. Hadoop, NoSQL, relational databases and other tools can be installed in virtual machines cluster. In this layer, according to a variety of business requirements, multiple data management solutions can be coexist, such as MySQL cluster, HBase or Hadoop Distributed File System (HDFS). Different data management solutions and diverse storage tools are applicable for different scale or types of data. For example, if local resource of single virtual machine is sufficient for data processing, it is not necessary to use YARN [17].

Above the virtualization layer, it is dataset processing layer, and Apache Flink is the most direct support for building this layer. It provides three layered APIs, and each API offers a different trade-off between conciseness and expressiveness and targets different use cases. ProcessFunctions is used to process individual events from one or two input streams or events that were grouped in a window and has fine-grained control over time and state. The DataStream API provides primitives for many common stream processing operations, such as windowing, record-at-a-time transformations, and enriching events by querying an external data store. Table API and SQL are used for unified stream and batch processing. Apache Flink features several libraries for

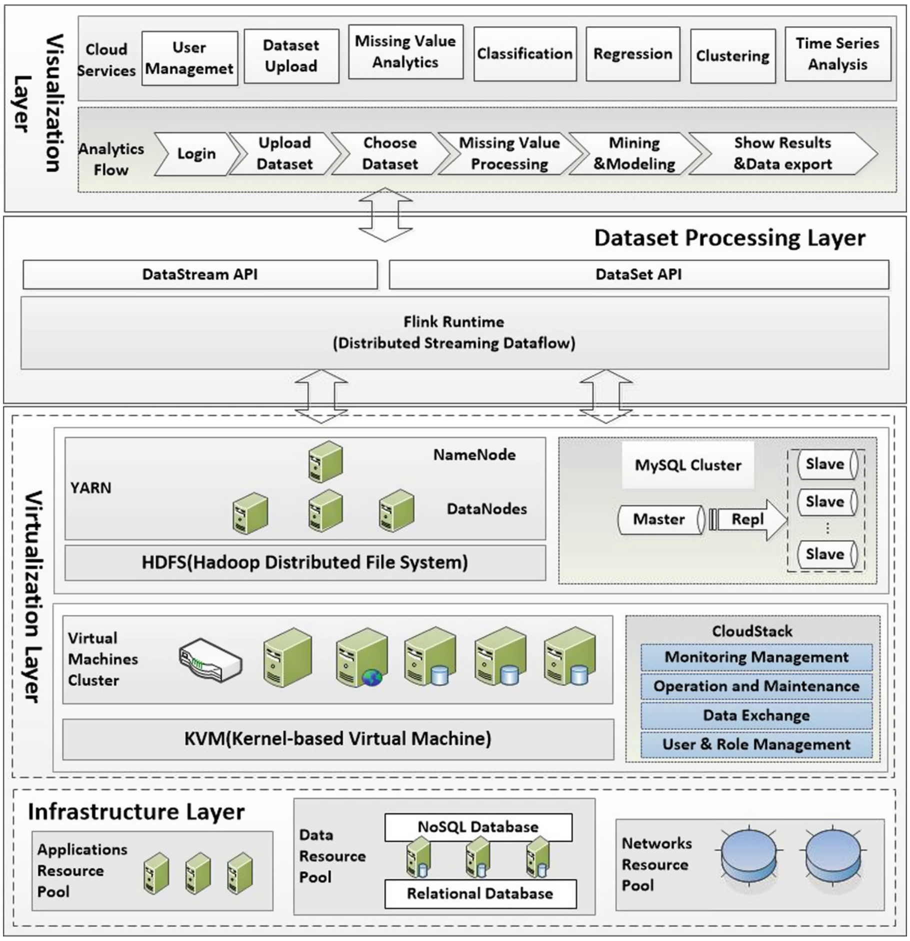

**Fig. 2.** The architecture of hydrological IoT data processing system

common data processing use cases. The libraries are typically embedded in an API and not fully self-contained. Based on such a rich API, we can implement many business functions and perform computations for hydrological IoT stream data at in-memory speed and at any scale. In addition, in order to implement an effective stream-first architecture and to gain the advantages of using Apache Flink, a common pattern is to implement a streaming architecture by using a message transport such as Apache Kafka [18], which can collect and deliver data from continuous events from a variety of sources (producers) and make this data available to applications and services that subscribe to it (consumers). Thus, having a message-transport system that decouples producers from consumers is better because it can support a micro-services approach and allows processing steps to hide their implementations, and provides them with the freedom to change those implementations.

In visualization layer, there are two main aspects to be considered: services and user interface. Firstly, based on the idea of service-oriented, many data query, data processing and analyzing are implemented into services. Secondly, the system provides WYSIWYG Web-based user interface for users.

## 4 Experiments and Discussion

The IoT dataset of real-time water level of Chuhe river from January 1, 2015 to June 30, 2017 is selected, with a total of 18,910,865 records. The experimental environment is a cluster made up of three same PC, and its configuration as follows: CPU is Intel(R) Xeon(R) CPU E5645@2.40 GHz dual-core 24 CPU; memory is Kingston DDR3 1333 MHz 8G, 500 GB SSD Flash Memory. Software tools are Ubuntu 6.04 64-bit, and Linux 3.11.0 kernel. For different storage mechanisms, our choice is MySQL 5.7.x, Kafka 1.1.0, MongoDB 2008 plus 3.6.3[19] and HBase 1.2.6 [20].

In the field of water conservancy informatization, traditional multi-tier architecture system based on Java EE or pure NoSQL databases are common and usually have the normal functions of finding specific value, finding extreme value and adding or deleting data. Therefore, the following experiments compare the differences between the proposed IoT data processing systems based on Apache Flink with traditional multi-tier architecture system based on Java EE or pure NoSQL databases under these common and daily operations.

The first experiment is to find out specific IoT value of water level, such as records of river water level above "5.5". It takes 8.41 s to access the MySQL database table in Java EE system. The same operation to access the IoT dataset in MongoDB directly takes 7.46 s. However, in our system based on Apache Flink, it only takes about 0.03 s to get the same results from HBase through Kafka.

In the second experiment, our purpose is to find out the extreme value in IoT data, such as finding out the records of the lowest water level in more than 70 hydrological monitoring stations. It takes 16.2 s to access the MySQL database table in Java EE system. The same operation to access the IoT dataset in MongoDB directly takes 10.3 s. In our system based on Apache Flink, it only takes about 0.07 s to finish the task from HBase through Kafka.

In the third experiment, the deletion operation for IoT data is tested. For large-scale IoT stream data, it does not need to be maintained for a long time, and it is often cleaned up after a period of time. For this reason, taking 5 million records as an example, we verify the effect. It only takes 3.22 s using the IoT data processing systems based on Apache Flink and HBase, far less than the 122 s needed to use MySQL logic in Java EE system and the 55 s needed to use logic for accessing MongoDB.

The experimental results are shown in Fig. 3. below, and it is not hard to see that the system based on Apache Flink platform makes full use of the parallelization mechanism, and significantly improves the execution efficiency of common operations.

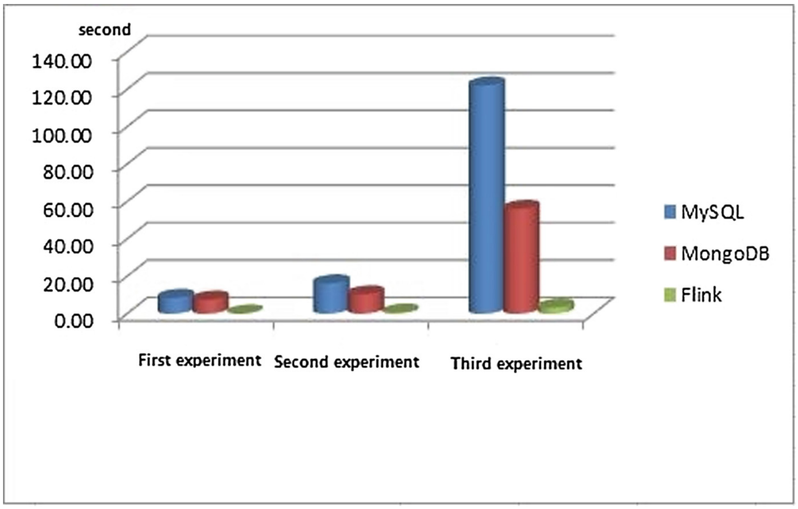

**Fig. 3.** Comparison of execution time in different experiments

## 5 Summary and Prospect

In this paper, we summarize the characters of big data of water conservancy domain, and then propose a hydrological IoT data processing system based on Apache Flink. Then, we analyze the performance of the proposed the hydrological IoT data processing system using the IoT data obtained in Chuhe river. By comparing the efficiency of common operations, our proposed system is significantly superior to the traditional application based on Java EE or pure NoSQL databases system.

In follow-up studies, we will focus on study the machine learning algorithms combined with the analysis of the hydrological IoT data on Apache Flink platform, especially for real-time analysis of stream data, and provide more support for flood control and drought control.

**Acknowledgement.** This work is partly supported by the 2018 Jiangsu Province Key Research and Development Program (Modern Agriculture) Project under Grant No.BE2018301, the 2017 Jiangsu Province Postdoctoral Research Funding Project under Grant No. 1701020C, the 2017 Six Talent Peaks Endorsement Project of Jiangsu under Grant No. XYDXX-078, and the Fundamental Research Funds for the Central Universities under Grant No. 2013B01814.

## References

1. Yu, R., Yang, X., Huang, J., et al.: QoS-aware service selection in virtualization-based cloud computing. In: Proceedings of 14th Asia-Pacific Network Operations and Management Symposium: Management in the Big Data and IoT Era, pp. 1–8. IEEE Computer Society (2012)
2. Walker, S.J.: Big data: a revolution that will transform how we live, work, and think. Math. Comput. Educ. **47**(17), 181–183 (2013)

3. Friedman, E., Tzoumas, K.: Introduction to Apache Flink: Stream Processing for Real Time and Beyond. O'Reilly Media, Sebastopol (2016)
4. Deshpande, T.: Learning Apache Flink. Packt Publishing, Birmingham (2017)
5. Feng, J., Xu, X., Tang, Z., et al.: Research on key technology of water big data and resource utilization. Water Resour. Informatiz. **8**, 6–9 (2013)
6. Helsel, D.R., Hirsch, R.M.: Statistical Methods in Water Resources. http://water.usgs.gov/pubs/twri/twri4a3/
7. Gong, H., Liu, W., et al.: Water resources data center construction based on big data. In: 3rd Water Conservancy Information and Digital Water Conservancy Technology Forum, pp. 243–248. Hohai University Press, Nanjing (2015)
8. Qin, X., Wang, H., Du, X., et al.: Big data analysis-competition and symbiosis of RDBMS and MapReduce. J. Software **23**(1), 32–45 (2012)
9. Lam, C.: Hadoop in Action. Manning Publications, Stamford (2011)
10. Bajaber, F., Elshawi, R., Batarfi, O., et al.: Big data 2.0 processing systems: taxonomy and open challenges. J. Grid Comput. **14**, 379–405 (2016)
11. Sakr, S., Liu, A., Fayoumi, A.G.: The family of MapReduce and large-scale data processing systems. ACM Comput. Surv. **46**(1), 10–11 (2013)
12. Zhao, S., Jiang, J.: Typical big data computing frameworks. ZTE Technol. J. **22**(2), 14–18 (2016)
13. Estrada, R., Ruiz, I.: Big Data SMACK: A Guide to Apache Spark, Mesos, Akka, Cassandra, and Kafka. Apress, New York (2016)
14. Zhang, P., Li, P., Ren, Y., et al.: Distributed stream processing and technologies for big data: a review. J. Comput. Res. Develop. **51**(Suppl), 1–9 (2014)
15. Sakr, S.: Big Data 2.0 Processing Systems: A Survey, pp. 74–89. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-38776-5
16. Liu, X., Iftikhar, N., Xie, X.: Survey of real-time processing systems for big data. In: Proceedings of the 18th International Database Engineering and Applications Symposium. Association for Computing Machinery, pp. 356–361 (2014)
17. Chintapalli, S., Dagit, D., Evans, B., et al.: Benchmarking streaming computation engines: storm, Flink and spark streaming. In: Proceedings of IEEE 28th International Parallel and Distributed Processing Symposium Workshops, pp. 1789–1792. IEEE Computer Society (2016)
18. Narkhede, N., Shapira, G., Palino, T.: Kafka: The Definitive Guide. O'Reilly Media, Sebastopol (2017)
19. Tiwari, S.: Professional NoSQL. Wiley, Indianapolis (2011)
20. George, L.: HBase: The Definitive Guide. O'Reilly Media, Sebastopol (2011)