Transactions on Computer Science and Intelligent Systems Research
ISSN: 2960-1800, eISSN: 2960-2238 | Volume 7
TPCEE 2024

Leveraging Big Data Technologies for Smart Grid Development: Frameworks and Implementation

Qichen Xue *

School of mathematics, Sun YAT-SEN University, Guangzhou, China

* Corresponding Author Email: xueqch@mail2.sysu.edu.cn

Abstract. As the urgency to ensure energy sustainability and environmental preservation grows in response to climate change and escalating energy demands, transitioning from conventional grid systems to technologically advanced systems become imperative. The proliferation and complexity of data generated during energy production, conversion, measurement, and calculation pose significant challenges to traditional power system management methods. Consequently, the development of a new advanced grid system, which is named smart grid, is proposed. This system incorporates big data technologies to effectively manage the vast amounts of data that were previously cumbersome and difficult to handle. This study introduces the background and fundamental concepts of smart grids and big data, providing a comprehensive overview of mainstream data processing frameworks such as Apache Spark, Apache Storm, and Hadoop. The characteristics and capabilities of these frameworks are demonstrated and compared. The paper summarizes the contributions of these big data technologies to the development of smart grids, identifies the difficulties encountered during implementation, and proposes Apache Flink as a solution to meet the evolving requirements.

Keywords: Smart Grid; Big Data; Data Processing Framework; Apache Flink.

1. Introduction

People's lives, work, and the operation and development of society all rely on electricity, so the importance of the power system to human society is self-evident. The power system involves four main functions: the generation, transmission, distribution, and control of electricity from the initial facilities to the final user. During these progresses, huge amounts of data are generated. The data collected from various resources and different ways often come in vast volumes and diverse structures. Extracting and calculating the data demands substantial resources and advanced technical skills.

Therefore, with the growth of the social population and the expansion of the city area, the traditional grid gradually has no capacity to adapt to the rapidly developing demand. Smart grid is the transformation and enhancement of the traditional power grid, which utilizes advanced information technologies to manage urban electricity in an interactive and intelligent organization and operation mode [1-4]. The U.S. Department of Energy defines a smart grid as: A smart grid is an extensively distributed, automated energy delivery network that enables two-way power and information flow, allowing for comprehensive monitoring from the power plant to the customer's personal appliances, including but not limited to heating systems, lighting, and refrigeration units. This advanced infrastructure incorporates distributed computing and advanced communication technologies, such as the Internet of Things (IoT) and machine learning algorithms, to provide real-time data and achieve immediate supply and demand balance at the device level [5]. It is generally believed that smart grids have good self-healing and adaptive capabilities, better situational awareness, economic efficiency, compatibility, and interactivity [6].

The main purpose of this study is to illustrate how advanced big data analytics platforms, equipped with immense computing power and flexibility, are utilized in smart grids to address the challenges of traditional grid systems. The paper reviews and summarizes the fundamental concepts and backgrounds of Apache Spark and Apache Storm, provides an analytical comparison of their core technical characteristics and examines their distinct applications within smart grids. The rest of this

article is organized as follows: Section II presents the features and applications of Apache Storm and Apache Spark. In Section III, the difficulties encountered and proposed solutions are discussed. Finally, Section IV provides the conclusion.

2. Method

2.1. Introduction of Big Data

As mentioned earlier, the main features of big data called 5V are volume, velocity, variety, value, and veracity [7-8]. Volume refers to the large amount of data generated during power collection, storage, and calculation. The data during the process is typically measured in 1024GB (TB) and 1024TB (PB) units. Speed means that a great number of new data is generated daily or even hourly, which requires high processing speed. The term "variety" encompasses a wide range of data types that require processing, including structured data, semi-structured data, and unstructured data such as weblogs, video, images, audio, and other forms of information. Value indicates that the data has research significance, but its density is low. The large amount of data combined with much useless information means that even though there is a massive number of data to calculate, the actual valuable data is few. Veracity refers to the accuracy and reliability of data, which has a significant influence on the analysis. The type of data is shown in the Fig. 1.

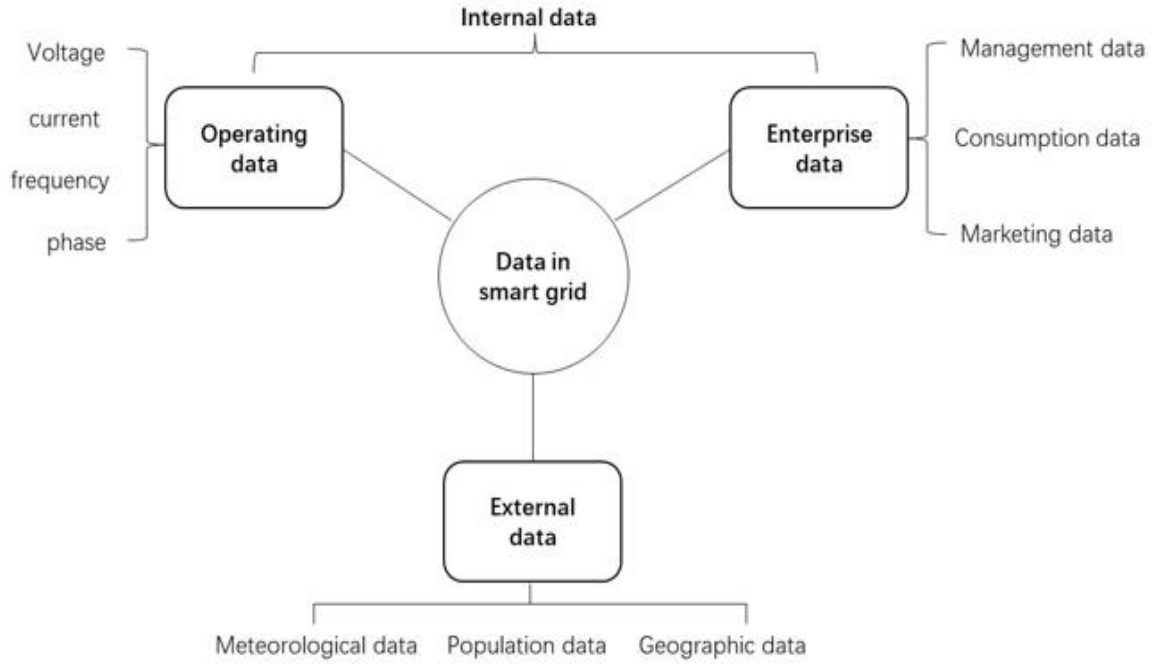

Figure 1. Types of Data in Grid.

2.2. Proposed Approach

The main technique currently used to deal with big data is distributed computing. Distributed computing is based on the principle that the task to be solved is divided into small parts and assigned to multiple computers for processing, thereby greatly improving efficiency. There are different types (Fig.1) of processing modes for computing tasks. This is because different distributed computing technologies complete the calculating tasks according to different implementation schemes. Batch processing and stream processing are two popular big data processing modes. The paper will illustrate the main ideas and salient features of Storm which is based on stream processing and Spark which is based on hybrid processing. The research process of this paper is shown in (Fig.2).

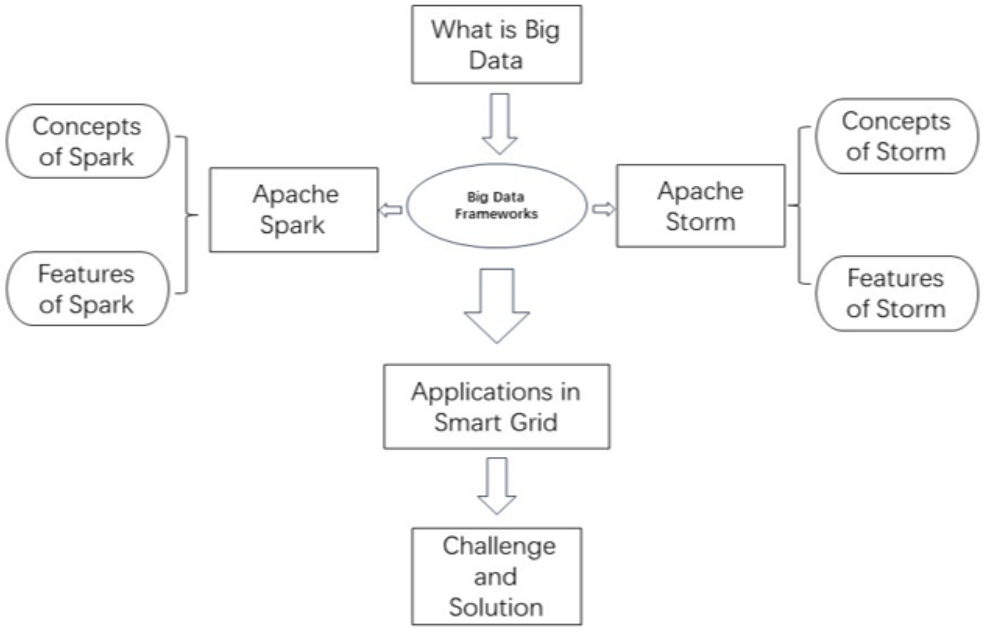

Figure 2. Research Process.

2.2.1. Introduction of Storm

Stream processing is a vivid description that refers to data being continuously read and processed like water flow, representing a real-time method of data processing. Storm is a typical big data platform based on stream processing. The architecture of Storm (Fig.3) mirrors that of Map Reduce (Fig.4). It is a classic batch-processing architecture that was initially put forward by Google to address specific problems related to search. Its simple structure and nice versatility enable it to handle large-scale data. However, many power grid faults or anomalies occur instantly, and the response time of batch processing may not be able to meet high-precision real-time processing. The millisecond-level response of stream processing is well-suited for such scenarios. The most notable feature of Storm is the use of a topology composed of different nodes. These work nodes can be roughly divided into two categories: bolt and spout. The spout is responsible for continuously reading from the data source and transmitting them in tuple form to bolt-type nodes. The bolt node will receive a tuple form of data transmitted from the spout and execute different functions according to established internal logic.

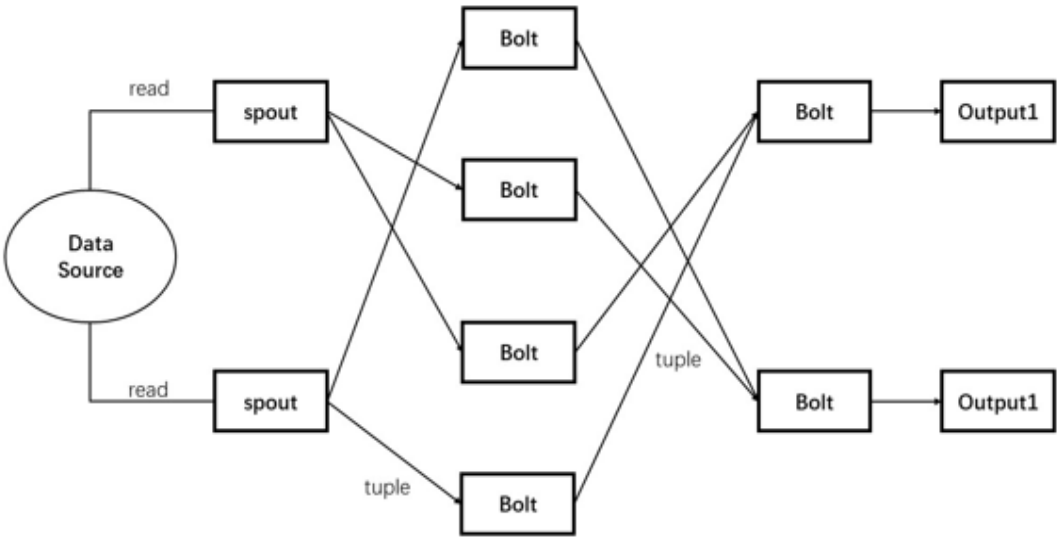

Figure 3. Architecture of Storm.

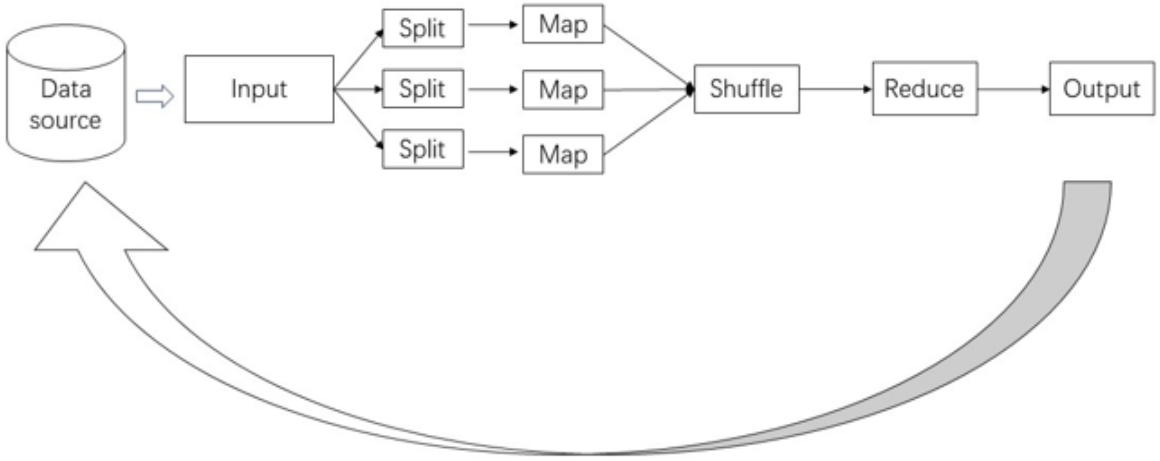

Figure 4. Architecture of Map Reduce.

Storm utilizes a master-slave architecture (Fig.5) similar to Hadoop. Distributed computing is completed by three main blocks: Nimbus, Supervisor, and Zookeeper. Nimbus is responsible for resource allocation and task scheduling to each working node and monitoring failed tasks. Zookeeper oversees the whole operation of the Storm cluster. The Supervisor is accountable for accepting tasks from Nimbus, starting and stopping the process for each worker that belongs to it.

Storm has some advantages compared with Hadoop. When a topology is submitted to the Storm cluster for the first time, the cluster performs an initialization for the topology, afterwards, there is no additional formatting work required. Because the topology only needs to be initialized once, Storm effectively avoids a large amount of time loss. Storm uses the Zookeeper to manage faults for each task. During the work process, if there is an exception, the topology will run a failure. Then Storm will restart consistently so that it can recover correctly, which strengthens the resistance to the emergency. Additionally, Storm uses Netty to transmit messages, ensuring that messages can be processed quickly. At the same time, Storm adopts an in-memory computing pattern. This pattern does not require file storage and can directly transmit intermediate calculated results through the network, preventing a large amount of time cost in data transmission between components.

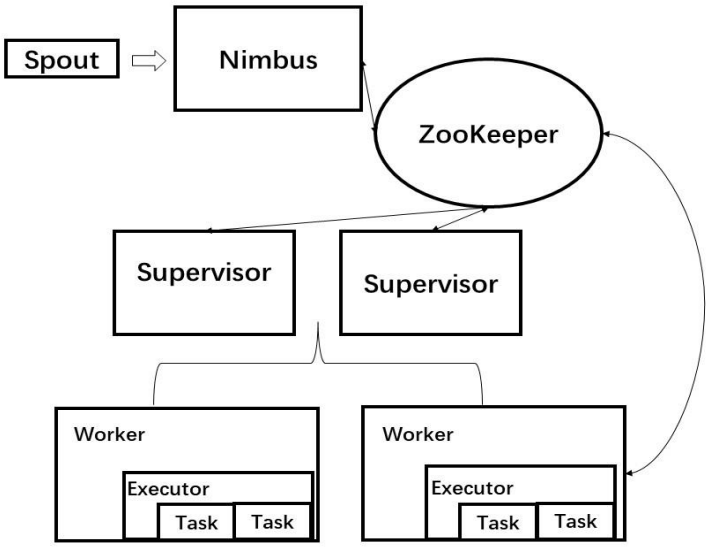

Figure 5. Framework of Storm.

2.2.2. Introduction of Spark

Spark is a typical framework based on a kind of data processing method called hybrid processing because it is compatible with both stream processing and batch processing. Actually, the processing mode of Spark is batch processing. This means the capability of stream processing of Spark

essentially is a type of batch processing framework using Spark Streaming. The mode of Spark Streaming is shown in (Fig.6). Spark streaming first separates the initial data flow into a great number of small batches. The principle to divide is the time of data, and then these batches are transformed into a Resilient Distribution Dataset (RDD) in a particular gap used for following processing. The rapid and consecutive batch processing over a short period allows RDD to manifest as a stream data flow. A salient feature of Spark is the use of RDD. RDD represents an immutable and partitioned dataset, in which internal elements can be calculated in parallel. An RDD consists of many partitions, and in Spark, the number of tasks being executed is determined by the number of these partitions. These sets are elastic, ensuring that if a dataset is lost, RDD can reconstruct the lost part, which improves the fault tolerance of Spark.

Besides, Spark Structured Query Language (SQL) is integrated into Spark to process structured data. This model provides a programming abstraction, that is the DataFrame, and can serve as a query engine for distributed SQL. Spark SQL can transform tasks into RDD through SQL and then submit them to the cluster for computing. This process is similar to Hive, which converts tasks into MapReduce through SQL likewise. By this method, Spark largely reduces the complexity of writing computing programs and makes the efficiency of execution higher than that of MapReduce. Spark also uses the Spark Machine Learning library (Spark MLlib) to provide a large number of machine learning methods, including classification, regression, clustering, etc. Spark MLlib also provides additional functions supporting model evaluation and data importing. It helps to combine machine learning algorithms with the distributed computing framework of Spark to improve performance and scalability.

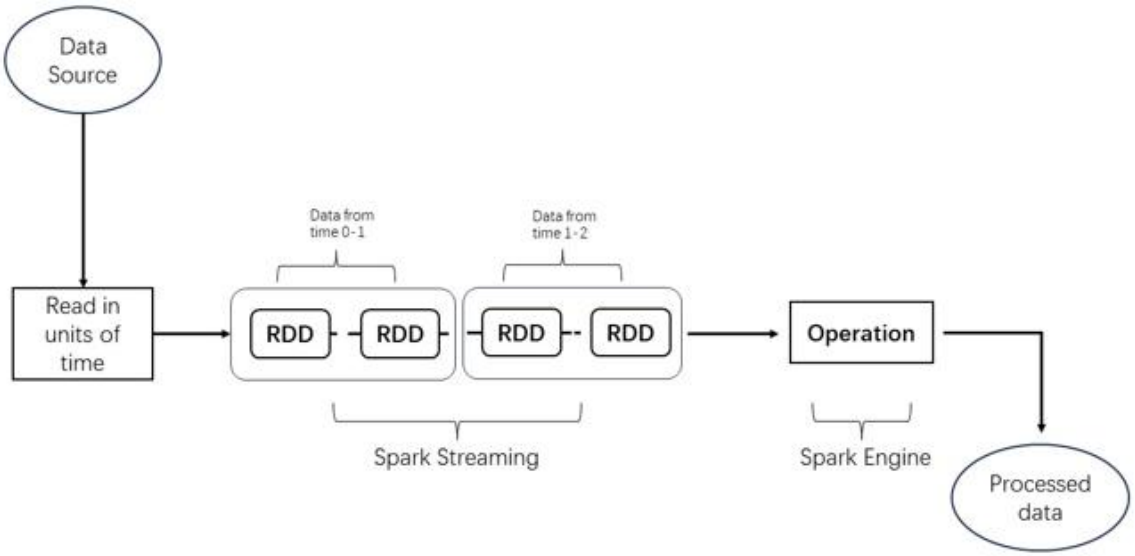

Figure 6. Process of Spark Streaming.

Apache Spark is currently the advanced iterative processing framework, capable of processing data across clusters and keeping it in memory. All iterations are complete once the data is read into the framework and written back. To compensate for the inefficiency of MapReduce due to network transmission and heavy disk Input/Output (I/O), memory is used in Spark to compute data, enabling Spark to complete queries faster and return calculated results in real-time. What's more, Spark provides more Application Programming Interface (API) at a higher level than Hadoop, and the running speed of the same algorithms in Spark is 10 to 100 times faster than in Hadoop.

2.2.3. Applications in Smart Grid

There is an application that uses Storm to implement real-time data encryption by designing different logic functions of the bolt in its topology . Spout reads external real-time data and streams it into the Topology. All of Storm's message-processing logic is encapsulated in Bolt. Spilt-Bolt is in charge of receiving tuples transmitted from Spout and diversifying tuples. The split tuples group is then sent to Bolts which encapsulate the encryption logic to execute. To ensure that the data is processed in a

consistent order, the system assigns a unique Identity (ID) to each tuple before execution. Once all the data is encrypted, it will be sent to the initial-sort-Bolt and sorted by ID for the first time. Finally, this once-sorted data is transmitted to the final-sort-Bolt for final sorting and output. The process is shown in (Fig.7).

Another case is the Phasor Measurement Unit (PMU). PMU is composed of a Global Positioning System (GPS) second pulse as a synchronous clock, which can be used to measure the voltage vector of each node in the transient process of the power system. As an important piece of equipment to ensure the operation of the power system, it has been widely used in the fields of dynamic monitoring, state estimation, system protection, regional stability control, system analysis and prediction. Traditional Distributed Control System (DCS) and Supervisory Control And Data Acquisition (SCADA) systems can be used for control and detection. However, interference changes in real-time data such as current occur in a very short time, and SCADA can only detect once every few seconds, which can cause some errors. Thus, integration with Spark or Storm improves fault tolerance.

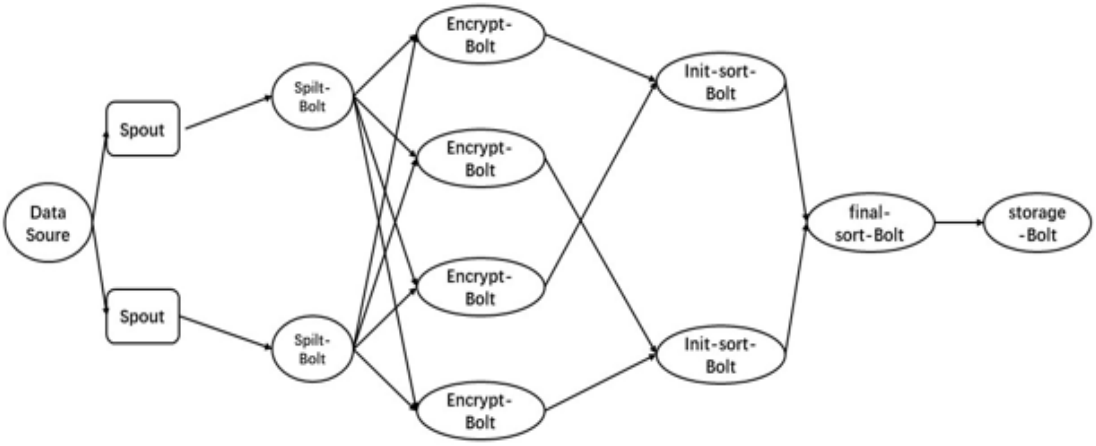

Figure 7. Process of Real-Time Data Encryption [9]

3. Discussion

3.1. Challenges

Even Though the benefits of big data technologies are many, some challenges and difficulties still need to be solved. Hadoop is often used to address business scenarios with high throughput, batch processing, and offline calculation results. If real-time data processing speed is required, Hadoop obviously does not meet the demand. As for Storm, it is notable for its extremely low latency, simple operation, maintenance, scalability, and high fault tolerance. Its pure real-time processing means one piece of data will be processed as it comes in. Hence a salient problem is that the data throughput of Storm is meager. It is also possible that the processing speed of the Bolt does not match the transmission speed of the Spout, which will result in data accumulation. When using Storm stream processing, splitting tasks properly and designing Topology is very important but complex. Compared to Storm, Spark has higher data throughput, making it more versatile. Within the Apache Spark framework, Spark Streaming can be perfectly integrated and coordinated with other departments such as Spark Core and Spark SQL, meaning that the intermediate data computed in real-time can be used seamlessly for batch processing, interactive query and other operations in the program. This feature greatly enhances the advantages and functions of Spark Streaming. While Spark streaming brings what is called stream processing, it is actually a mode of batch processing with second-level latency. However, this pattern may not perform quickly enough in situations that demand very low latency.

3.2. Solution

To meet the requirements of both low latency and high throughput, Flink was launched. Flink is essentially a stream processing framework with the compatibility to work on batch data. Flink

consumes less and has higher throughput than Storm, has lower latency than Spark, and is easier to program. Apache Flink provides a special mechanism to improve fault tolerance which can consistently restore the state of data flow applications. During the calculation process, Flink generates consistent snapshots of distributed data flow and the state of operators by this mechanism. When a fault occurs, the system can roll back to the location of these snapshots and start working again.

In the field of stream processing, incremental calculations on unbounded streams are often assessed over continuously operating logical views, called windows. windows play an important role, but most frameworks use processing time for computing in windows. The processing time is the current time of the system when the event is transmitted to the framework to be processed. Flink can support windows based on event time, which means using the time when the event is generated. This event-driven mechanism enables Flink to calculate accurate results even if the event arrives in an out-of-order manner, ensuring the original time sequence of the event.

4. Summary

This study introduces the concept of the Smart Grid, highlighting the significance of big data and its integration with grid systems. It provides a comprehensive examination of two mainstream big data frameworks—Apache Spark and Apache Storm—adopted for different applications within the Smart Grid. Apache Spark is presented as a hybrid processing framework capable of handling both batch and stream data through Spark Streaming, while Apache Storm is described as an efficient stream processing framework with very low latency and high throughput. Despite their respective limitations, neither framework is universally superior; instead, each is suited to specific scenarios. As Smart Grid technology continues to advance, the integration of sophisticated big data analytics platforms like Apache Flink holds promise for revolutionizing power system management, enhancing resilience, efficiency, and adaptability in response to dynamic demands and challenges. Future research will focus on the feasibility of applying Apache Flink within Smart Grids, investigating whether its application can improve the universality and efficiency of power grid systems while also balancing economic consumption.

References

[1] G. Dileep. A survey on smart grid technologies and applications. Renewable energy, 146 (2020), 2589-2625.

[2] X. Fang, S. Misra, G. Xue, et al. Smart grid—the new and improved power grid: A survey. IEEE communications surveys & tutorials, 14(4) (2011): 944-980.

[3] M.L. Tuballa, M.L. Abundo. A review of the development of Smart Grid technologies. Renewable and Sustainable Energy Reviews, 2016(59), 710-725.

[4] A. Shobol, M.H. Ali, M. Wadi, et al. Overview of big data in smart grid. International Conference on Renewable Energy Research and Applications, (2019), 1022-1025.

[5] A. Bari, J. Jiang, W. Saad, et al. Challenges in the smart grid applications: an overview. International Journal of Distributed Sensor Networks, 10(2) (2014), 974682.

[6] K. Moslehi, R. Kumar. A reliability perspective of the smart grid. IEEE transactions on smart grid, 1(1) (2010), 57-64.

[7] R. Shyam, B.G. HB, S. Kumar, et al. Apache spark a big data analytics platform for smart grid. Procedia Technology, 21 (2015), 171-178.

[8] H. Ali El-Sayed Ali, M.H. Alham, D.K. Ibrahim. Big data resolving using Apache Spark for load forecasting and demand response in smart grid: a case study of Low Carbon London Project. Journal of Big Data, 11(1) (2024), 59.

[9] S. Zhang, J. Sun, B. Wang. A study of real-time data encryption in the smart grid wide area measurement system based on Storm. International Conference on Machinery, Materials and Information Technology Applications. Atlantis Press, (2015), 532-537.