EasyChair Preprint

№ 9544

Distributed Computing & Smart City Services

Sudirgha Chakma, Annajiat Alim Rasel and Md Sabbir Hossain

January 5, 2023

Distributed Computing & Smart City Services

Sudirgha Chakma
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
Sudirgha.chakma@g.bracu.ac.bd

Annajiat Alim Rasel
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
annajiat@bracu.ac.bd

MD Sabbir Hossain
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
Sabbir.hossain@bracu.ac.bd

Abstract: The widespread grow of big data and the evolution of Internet of Things (IoT) technologies enable cities to obtain valuable intelligence from a large amount of real-time produced data. In a Smart City various IoT devices generate data continuously which needs to be analyzed within a short period of time, using some big data technique. In this paper, we examine applicability of employing distributed stream processing frameworks at the data processing layer of Smart City and appraising the current state of their adoption and maturity among the Smart City use cases. Our experiments focuses on evaluating the performance of three SDPSs, namely Apache Storm, Apache Spark Streaming, and Apache Flink. According to our obtained results, choosing proper framework at the data analytics layer of a Smart City, depends on characteristics of the target IoT applications. Finally, we present a category of applications that suit each framework.

Keywords
Smart City, Internet of Things, Big Data, Distributed Computing.

1. INTRODUCTION

Now smart cities are a new amount of urbanization of the digital services. The smart city concept is a way to make digital solution and more inclusive in transportation, education, advanced healthcare and many other aspects of our daily life. Smart cities can promote our lifestyle, many innovation and others technology that makes smart cities very efficient. For example wind turbines are used to get power in street light as well as our home also. By the using of bicycles is very essential for us to ignore the environment pollution all around the city. With unidentified data and services the transformation to digital cities often results in applications and data soils.
In a Smart City generated data normally has the following characteristics:

• **Large volumes of data**: the amount of real-time generated data by different applications in a Smart City can be in order of terabytes.
• **Heterogeneous data sources**: in Smart Cities, the data sources are diverse; for example, there are many sensors data, RFID data, cameras data, human generated data, and so on.
• **Heterogeneous data types**: collected data by different devices are different in format, packet size, required precision and arrival time.

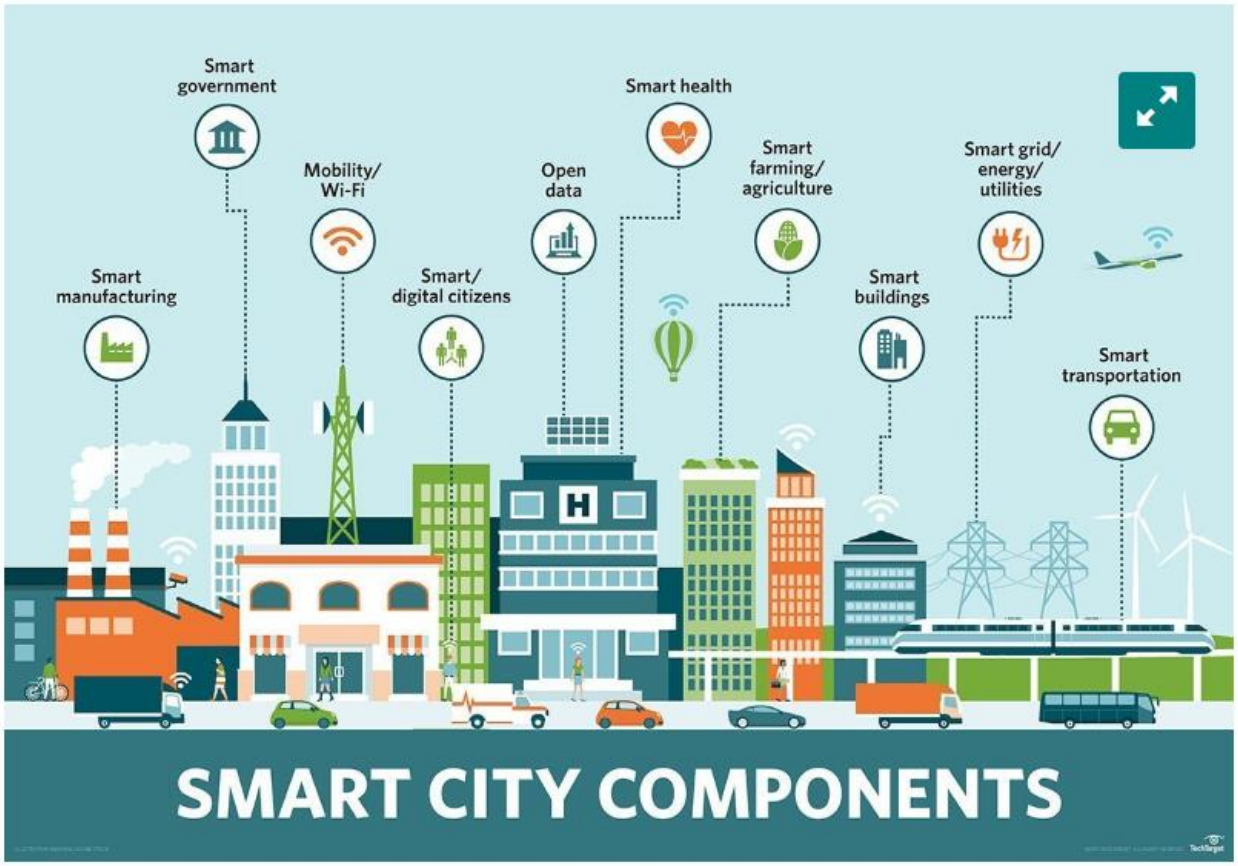
1.1 Smart city

A smart city is a municipality that uses information and communication technologies to increase operational efficiency, share information with the public and improve both the quality of government services and citizen welfare. A smart city success depends on its ability to form a strong relationship between the government including its bureaucracy and regulations and the private sector. The relationship is necessary because most of the work that is done to create and maintain a digital, data-driven environment occurs outside of the government. Suveillance equipment for busy streets could include sensors from one company, cameras from another and a server from yet another.

1.2 Features of a smart city

Any area of city management can be incorporated into a smart city initiative. For example is the smart parking meter that uses an application to help drivers find available parking spaces without crowded city blocks. The smart meter also enables digital payment so there is no risk of coming up short of coins for the meter.

Also in the transportation area, smart traffic management is used to monitor and analyze traffic flows in order to optimize streetlight and prevent roadways from becoming too congested based on time of rush hour schedules.



Smart city technology is increasingly being used to improve public safety from monitoring areas of high crime to improving emergency preparedness with sensors. For example smart sensors can be critical components of an early warning system before droughts, floods, landslides or hurricanes.

1.3 System Architecture for smart city

The system structure that integrates big data into Smart City can be considered as bellow. Layers in this architecture are similar to IoT system architecture layers and can be divided as follow:

1. Device layer: set of sensors, RFID, cameras, and other devices that capture data continuously and are connected via a network.

2. Data collection layer: a collection of unstructured data captured by devices and stored in big data store systems such as Cassandra, Hbase, MangDB, and Redis.

3. Data processing layer: In this layer, the stored data are processed using batch or stream distributed processing engines like Hadoop, Spark, Storm, and Flink.

4. Application service layer: Here many applications such as intelligent traffic management, water and electricity monitoring, disaster discovery, fraud detection and web display analysis are provided. In this layer people and machines directly interact with each other.

The goal of this survey is to examine applicability of employing distributed computing processing frameworks at the data processing layer of smart city.

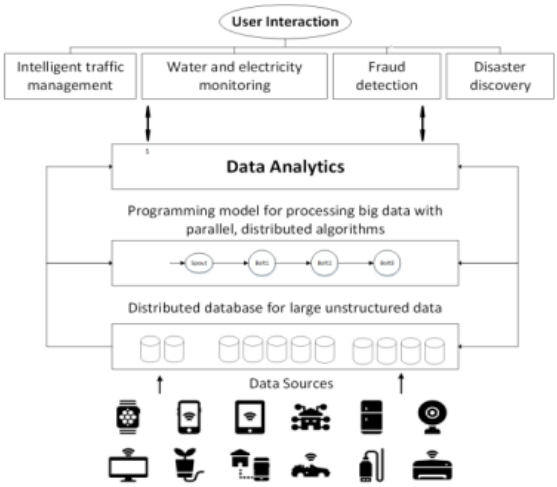


Fig 1: System architecture for smart city

2. Evaluation

There are many distributed stream processing frameworks which can be employed as the data analytics layer of IoT applications in Smart Cities. Apache Storm, Apache Spark, and Apache Flink are the most popular DSPFs for stream processing and in this section we have evaluated these three open source and community driven frameworks. To compare the performance of DSPFs we measured end-to-end latency and throughput as the major metrics of evaluation. Here end-to-end latency is the spent time for a tuple to be completely processed by framework and throughput is the number of tuples that frameworks can process in a given amount of time.

2.1 Experimental Setup

For our experimental evaluation, we have set up a cluster with 9 machines. Table 1 shows the specification of master and worker nodes. Machine one is the master node which runs job manager and its corresponding demons, and all employed worker nodes have same characteristics and are from type two. In all experiments we have used Apache Storm 0.10.0, Apache Flink 1.2.0 and Spark 2.2.1 on top of Ubuntu 16.04 operating system.

Table 1. System Characteristics

Node Type	Machine 1 (master)	Machine 2 (worker)
Processor	Intel Core i7	Xeon E5-2650
Memory	8 GB	4 GB
Network adapter	1 Gb	1 Gb
Operating System	Ubuntu 16.04	Ubuntu 16.04

2.2 Benchmark Application

The benchmark program is a sample IoT application. In this program each incoming tuple is processed step by the following operations respectively.

Source: This component reads tuples from Kafka message broker and prepares them as standard data units according to the DSPF's data processing model.

Deserialize: Divides the input JSON string to some meaningful fields.

Filter: Filters out irrelevant tuples based on their type. Projection: Remove unnecessary fields.

Join: Joins tuples by a specific field with its associated information of another field.

Count: Take a windowed count of tuples per joined field and store them.

2.3 Scalability Evaluation

To observe behavior of DSPFs under different loads we have run benchmark with a range of input rates. The rate at which data source emitted data tuples into the processing system is varied from 20k tuples/sec to 560k tuples/sec. For each input rate, benchmark application is executed for 100 minutes and end-to-end latencies are measured. To make sense about constancy of DSPFs the 99th percentile latency is calculated at each rate and the values are illustrated in figure 10. By placing resource utilizations of all DSPFs under scrutiny, we realized Flink is more network intensive than other frameworks while its CPU usage is less than both Storm and Spark. As we can see in figure 2 and 3 Flink has stable performance while input rate is increasing, but at certain rate the network gets bottleneck and its latency become worse than both Storm and Storm no-Ack. These results say while network resources do not get bottleneck Flink provides more stable response time and its 99th percentile latency value says its better solution for hard real-time applications. On the other hand, Spark Streaming latency strongly depends on input rate. It would not be good choose for application with variable data load like network monitoring. With acking disabled, Storm has better performance and provides more reliable response times at high throughput. However, in this conditions the ability to handle failures is disabled. The last experiment is related to scaling ability of intended frameworks. Here we increased cluster size by adding more worker nodes from 2 to 8. Figure 4 and 5 shows scale out a distributed system for smart city.

Looking at these graphs we can see Flink has worse scalability and its latency and throughput have few improvements when the number of worker nodes is

increased. Storm and Spark beat Flink in terms of latency and throughput respectively for bigger cluster. Both Storm and Spark have near linear behavior in terms of scalability but Storm scales even better than Spark and behaves almost linear when there is no acking mechanism.

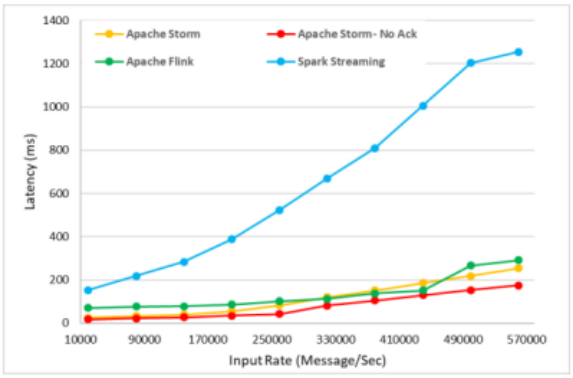


Fig 2: Average latency comparison

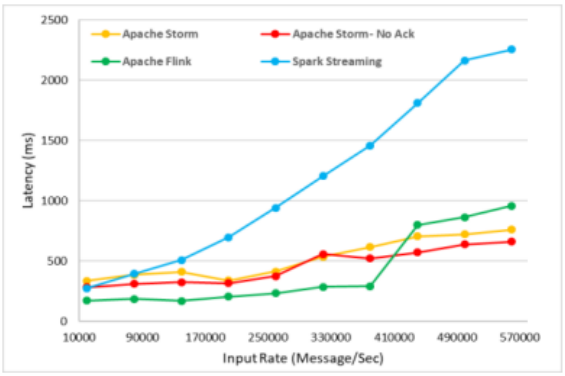


Fig 3: Latency comparison

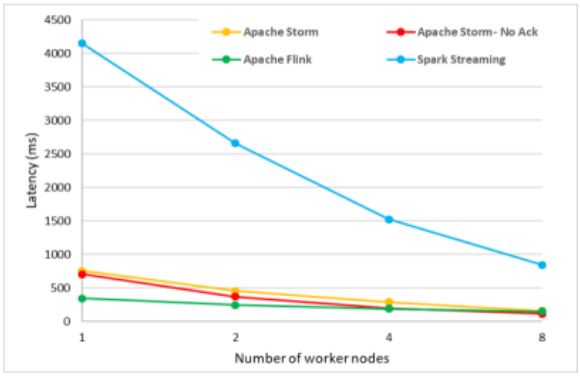


Fig 4: Average latency comparison for different number

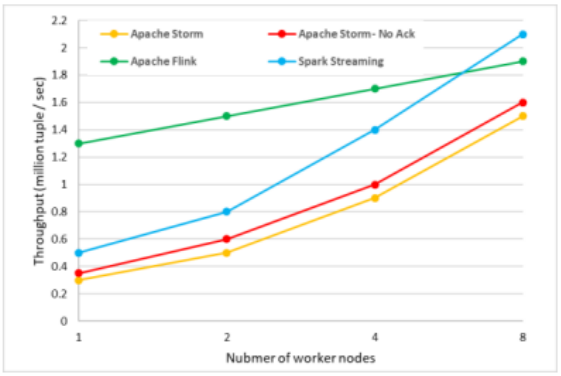


Fig 5: Comparison for different number of worker nodes

3. Conclusion

Collections of large amount of IoT devices and objects are producing huge amount of data in Smart Cities which requires being processed immediately. Big data analytics tools have the capacity to handle large volumes of data generated from IoT devices that create a continuous stream of information. There are plenty of big data processing platforms which each one is designed for special purposes. At the age of IoT and Smart Cities it is interesting to compare the behavior of available distributed stream processing frameworks and examine the applicability of employing them to process high volume of data generated in Smart Cities.

References

1. "Apahe Kafka: A distributed Streaming Paltform." [Online]. Available: http://kafka.apache.org/. [Accessed: 12-Feb-2018].
2. F. Chen, P. Deng, J. Wan, D. Zhang, A. V. Vasilakos, and X. Rong, "Data mining for the internet of things: Literature review and challenges," Int. J. Distrib. Sens. Networks, vol. 2015, no. i, 2015.
3. M. Zaharia et al., "Apache Spark: a unified engine for big data processing," Commun. ACM, vol. 59, no. 11, pp. 56–65, 2016.
4. G. Hesse and M. Lorenz, "Conceptual Survey on Data Stream Processing Systems," 2015 IEEE 21st Int. Conf. Parallel Distrib. Syst., pp. 797–802, 2015