International Journal of Engineering & Technology, 7 (1.1) (2018) 237-239

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

A distributed big data library extending Java 8

MD. A R Quadri [1], B. Sruthi [2]*, A. D.SriRam [3], B. Lavanya [3]

[1] *Koneru Lakshmaiah Educational Foundation, Vaddewsaram*
[2] *Department of computer science and engineering*
**Corresponding author E-mail: sruthibodala@gmail.com*

Abstract

Java is one of the finest language for big data because of its write once and run anywhere nature. The new release of java 8 introduced few strategies like lambda expressions and streams which are helpful for parallel computing. Though these new strategies helps in extracting, sorting and filtering data from collections and arrays, still there are problems with it. Streams cannot properly process with the large data sets like big data. Also, there are few problems associated while executing in distributed environment. The new streams introduced in java are restricted to computations inside the single system there is no method for distributed computing over multiple systems. And streams store data in their memory and therefore cannot support huge data sets. Now, this paper cope with java 8 behalf of massive data and deed in distributed environment by providing extensions to the Programming model with distributed streams. The distributed computing of large data programming models may be consummated by introducing distributed stream frameworks.

Keywords: *Big Data; Distributed Computing; Java; Streams.*

1. Introduction

In general, big data is used for computing large data sets and complex applications which are difficult to process using normal database processing. Previously in java, it is difficult to execute big data. Now because of new models that are introduced in java8 has simplified data parallel computing. Thus, new models like streams **Error! Reference source not found.** and lambda expressions **Error! Reference source not found.** facilitate us to permit view data processing through pipeline of operations. And java's lambda expressions a primary step to functional programming. Still these recent abilities like lambda expressions and streams lacking much necessary functionalities like processing very large sets and allowing computations over multiple machines. Now we propose the extensions to allow their usage in Big Data systems and proposed a methodology to work in distributed environment.

2. Back ground

Java is the foundation language for many popular big data frame works, like Hadoop **Error! Reference source not found.** and spark. Programmers can view the data processing by pipeline of operations through java8 **Error! Reference source not found.** streams. Also, Lambda expressions simplify functional programming.

Java8 streams entitle a sequential flow of objects from a source, which supports huge operations. Java 8 Streams provide a new data-parallel computation for programming models on a single machine. New stream API can make pipeline processing. On multi core hardware it can be able to parallelize the computation on various threads to speed up execution.

Lambda expressions can add functional processing ability to Java, allowing brief instantiation of functional interfaces **Error! Reference source not found.** instead of using unknown class syntax. These are used to a detailed degree in Java 8 **Error! Reference source not found.** Streams, helping its code more readable.

2.1. Limitations of streams

- Streams are restricted to operations that are within a single system **Error! Reference source not found.**. Also distributed computing over multiple machines is not possible. Since java is build on java's executer framework where there is no idea of distribution.
- The Stream takes Arrays, collections, Input/output resources as input source. The data will be stored in memory. So, these cannot aid for large datasets.

2.2. Map reduce functionality in hadoop

Hadoop **Error! Reference source not found.** using the HDFS (Hadoop Distributed File System) distributes the required data over the cluster. And the above happens before map reduce computation. Map reduce can be done by implementing mappers and reducers which run on a node in the cluster. Map Reduce computation **Error! Reference source not found.** consists of few stages and they are explained below.

Map Stage: The input stored in the HDFS is passed to mapper **Error! Reference source not found.** line by line. The data is processed by mapper also it creates many key value pairs as a result.

Reduce stage: Reduce stage is blended of both shuffle and Reduce stage. Reducer processes the key-value-list pairs that come from the mapper. And the output set will be stored in HDFS.

The functions of Map Reduce, map and reduce are both defined with reference to key value pairs. Thus, the list of key value pairs is transformed in to list of values by MapReduce framework. There ought to be a means of connection between the processes

accomplishing Mapper and reduce phases for MapReduce of distributed system.

2.3. Java 8 streams

Stream represents a sequential flow of objects from a source, which aids heap operations. Streams provide a set of constituents of specific kind in serial manner **Error! Reference source not found.**. Stream computes them on demand. With java 8, interface of collections have two strategies to produce streams.
Stream () - The function stream is used to create sequential stream that is one stream followed by other stream which composed of distinct elements
Parallel Stream () - They allow you to implement parallelism using collection as its source.
After retrieving the streaming data from the source, pipelining will be done. Almost all the operations of streams return stream itself so that result is pipelined. These operations are known as intermediate operations and their strategy is to grab input, process them and return output. Stream operations perform iterations internally on the source elements providedvbut explicit iteration is required in collections which is contrast to prior one. A stream has to be created again to use the data source as they are traversed only once.

3. Related work

As mentioned in **Error! Reference source not found.** for each stream type there will be numerous count of reduction operations. At each local data element reduction is performed. First node gets the result which sends to remaining nodes. At last same values are returned by every node. Though our method doesn't let map reduce to produce large amount of data as result instead it combines and gives single value. Hadoop **Error! Reference source not found.** specified iterative methods of data processing for big clusters, which in real big task for map reduce. A distributed frame work for cluster computing is given in **Error! Reference source not found.** which vary our approach. The data parallel computing capability of streams **Error! Reference source not found.** helped our work to move forward. Twister **Error! Reference source not found.** included an iterative approach for data where it also included data cache over the functionality of map reduce. Hadoop **Error! Reference source not found.** performs operations through stages, which works over a single node. And no cluster operations are possible. There is no distribution of data across systems. HDFS **Error! Reference source not found.** is used by both hadoop and spark to get the data. They can use the common data source. The arrays and collections of java preserve data in memory **Error! Reference source not found.** which are incapable of storing large data.

4. Proposed work

As to compute big data in java, our initiate expansions can fulfil the present programming model to work in a distributed environment. The distributed computing requirements to big data are made easy through distributed data framework. And the below strategies are followed to expose what we proposed.

4.1. Distributed streams

The distributed streams can allow operations to run simultaneously on different nodes at the same time since the data is already partitioned across nodes. Distributed stream **Error! Reference source not found.** allows data parallelism by using pipeline of operations at different parts of dataset by replicating them. There are no limits in Distributed stream for programming paradigm. Pipelining is the foremost functionality of streams.

4.2. Grouping of compute nodes

A process is to be created for finding and grouping compute nodes. This process is helpful in taking decisions like which node to send data to. So, two classes are defined they are compute node class and compute group class. Compute node class refers to node computation in cluster and compute group class refers to variety of compute nodes which can be seen in Fig 1. The compute group class takes the nodes as a group of them. Get Cluster method is invoked to get the entire cluster. After grouping the nodes collection is given as source.

4.3. Distributed collections

The information that is partitioned across the collection is wrapped by the distributed collection. A java collection is extended by distributed collection interface by the below

- The stream and parallel stream which are overridden will now return distributed streams.
- A new distributed collection is created by wrap method by grouping together normal collections across nodes.

5. Distribution of data

The isolated parallel machines are instantaneously replaced for authoritative distributed ones. As we face many problems in stimulating from a parallel program designed for mutual memory architecture to a distributed memory. For that purpose, the information may be shared in a way to benefit the locality and reusability of data. There is no charge for distributed stream model to execute all the participating nodes similar with the pipeline evaluation of operations. Pipeline can read the information from the input nodes and filtered on independent set of computes nodes and accumulate on different output nodes. This is important for splitting the stream.

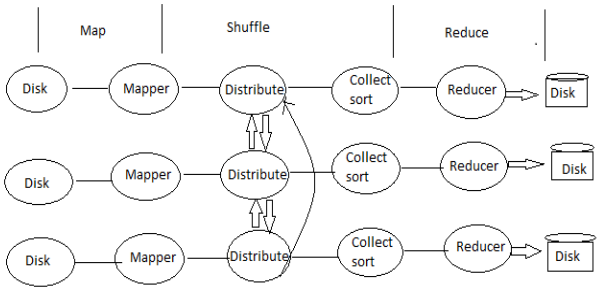

Fig. 1: Distribution of Data across Nodes.

5.1. Propagation

In cluster the propagation of data is possible by providing few standard interfaces which permit user to intimate the allotment of data. This method helps data transfer between one batch to another compute batch. The same data elements that are appeared in the nodes will be present in the new distributed stream that is returned by distribute operation. The split and join operations are also used on distributed streams. To maintain data locality the data cannot be transferred among the nodes by these operations.

6. Observations

The implementation of distributed streams is notably faster when compared to hadoop working. Therefore, the execution time for

distributed streams is also less and maximum of the time depends on the range of the input. Also, the system usage for distributed streams is more but it stills runs fast alike other frameworks. And the above observations reflect the competence of distributed streams.

7. Summary

In this paper we have proposed distributed stream library that is extending java8. We identified some inabilities of java8's new release. And specified how the distributed operations can be drifted out on multiple systems. Also shown hadoop's map reduce functionality in the proposed distributed stream library, given methodologies for the processing of huge datasets. Proposed extension of streams to distributed streams. The proposed method starts with distributed stream framework which distributes work over nodes. The computing an grouping of nodes before the allocation of nodes. Then about the distributed collections which take large data sets from local disk and next computation of distributed data.

8. Conclusion

The first thing we have written in this paper is about the Distributed streams using Java8 and comparison with the Hadoop and Map Reduce Framework of Computing. The ample majority of Java8 programming model was straight forwardly applicable to distribution. However, the fundamental difficulty watched was that short out assessment is exceptionally inefficient when distributed. As needs be, another operation must be added to the Java model to consider this. Something else, the proposed system is a drop-in trade for existing Java 8 code. What's more, pipelines have been made more flexible in Distributed Streams, taking into account more expressiveness in the programming model. All the more by and large, this work took generally at taking an identical programming model (Java 8 Streams) and forming it dispersed. This gives in reverse similarity and enables software engineers to incrementally streamline segments of their code. This is rather than most Big Data [13] programming models which begin as distributed models **Error! Reference source not found.**. Because of slight changes in functions, the optimization for implementations is possible. We assume except the network usage this approach is prior in using for distributed computing.

References

[1] Cho JH, Chang SA, Kwon HS, Choi YH, KoSH, Moon SD, Yoo SJ, Song KH, Son HS, Kim HS, Lee WC, Cha BY, Son HY & Yoon KH (2006), Long-term effect of the internet-based glucose monitoring system on HbA1c Reduction and glucose stability: a 30-month follow-up study for diabetes management with a ubiquitous medical care system. *Diabetes Care* 29, 2625–2631. https://doi.org/10.2337/dc05-2371.

[2] Fauci AS, Braunwald E, Kasper DL & Hauser SL (2008), Principles of Harrison's Internal Medicine, Vol. 9, 17thedn. *McGraw-Hill*, New York, NY, pp.2275–2304.

[3] Kim HS & Jeong HS (2007), A nurse short message service by cellular phone in type-2 diabetic patients for six months. *Journal of Clinical Nursing* 16, 1082–1087. https://doi.org/10.1111/j.1365-2702.2007.01698.x.

[4] Lee JR, Kim SA, Yoo JW & Kang YK (2007), The present status of diabetes education and the role recognition as a diabetes educator of nurses in korea. Diabetes Research and Clinical Practice 77, 199–204. https://doi.org/10.1016/j.diabres.2007.01.057.

[5] McMahon GT, Gomes HE, Hohne SH, Hu TM, Levine BA & Conlin PR (2005), Web-based care management in patients with poorly controlled diabetes. *Diabetes Care* 28, 1624–1629. https://doi.org/10.2337/diacare.28.7.1624.

[6] Thakurdesai PA, Kole PL & Pareek RP (2004), Evaluation of the quality and contents of diabetes mellitus patient education on Internet. *Patient Education and Counseling* 53, 309–313. https://doi.org/10.1016/j.pec.2003.04.001.