Asian Journal of Information Technology 15 (16): 2999-3003, 2016
ISSN: 1682-3915

A Study on Non Java Options for Mapreduce Programming with Hadoop

[1]C. Ranichandra and [2]B.K. Tripathy
[1]School of Information Technology and Engineering,
[2]School of Computer Science and Engineering, VIT University, Vellore, Tamil Nadu, India

Abstract: Storage and processing of intensively growing data has become easier with the invent of cloud computing. Computing paradigms have evolved from the era of centralized to distributed, distributed to grid and grid to cloud. Parallel programming models to process large data and with utilization of more cpus have got a new color with the introduction of mapreduce programming model. Hadoop, a framework for handling millions of data in clusters of computer uses MapReduce programming model. Users prefer java for their MapReduce jobs as Hadoop is written in Java. Here, we discuss the MapReduce programming approach and other approaches/languages that can be used to write MapReduce jobs.

Key words: Cloud, hadoop, mapreduce, streaming, pipes, pig, Hive, Jaql, cascading, dataflow

INTRODUCTION

With the invent of distributed computing, data processing can be done with databases stored at different locations across boundaries by fragmentation and distributed query processing policies (Ozsu and Valduriez, 1991). Distributed databases can be accessed by database links. This led to the sharing of data with computers located physically far apart like www.amazon.com server and www.amazon.in server. Users can query the sites without the awareness of where it is located.

In order to increase the computing power by utilizing the idle resources SETI (Search for Extra Terrestrial Intelligence) project was developed in early 2000 which paved the way for a new computing era called grid computing.

In a grid, data from different computers with different storage model, software and platform which are in different administrative domains can be easily shared by researchers (Taniar *et al*., 2008). Heterogeneous databases can be shared in a grid environment by OGSA-DAI which fits with Globus Toolkit.

Ubiquitous, convenient, on-demand access of resources are provisioned and released with minimal effort of service provider interaction in the new era of cloud computing. Resources are shared in a cloud by providing it as a service (Mladen, 2008). Services include software as a service (saas), platform as a service (paas), infrastructure as a service (Iaas) and last but not the least database as a service (daas). On demand, unstructured data can be easily deployed and handled by No SQL databases. There are lot of NoSQL databases that work mostly on key-value concept and well suited for cloud environment (Chandra *et al*., 2012).

HADOOP IN CLOUD

Hadoop was developed by doug cutting which has its origin in Apache Nutch an open source web search engine (White, 2012). Hadoop can process large distributed data across clusters of computers with simple programming model. Hadoop comes with Distributed File System (HDFS) for handling large data that can be partitioned across many nodes in the cluster. The HDFS works on the principle of one namenode and multiple datanodes. Namenode manages the file system namespace and datanodes are the home for the files.

Mapreduce: Mapreduce is a framework for processing and managing large-scale datasets in a distributed cluster. It provides transparencies such as data distribution and scheduling. Mapreduce works on a simple model-key value pair, consisting of map and reduce functions (Sakr *et al*., 2013). Map function groups the data, sorts and send to reducer. Reduce function merges and consolidates the result from map as shown in Fig. 1.

Suppose, we have three datanodes (three slaves) and one namenode (one master). The namenode runs the job tracker process and controls the flow of a map reduce process. The datanodes run the task tracker to complete the subtask.

Corresponding Author: C. Ranichandra, School of Information Technology and Engineering, VIT University, Vellore, Tamil Nadu, India

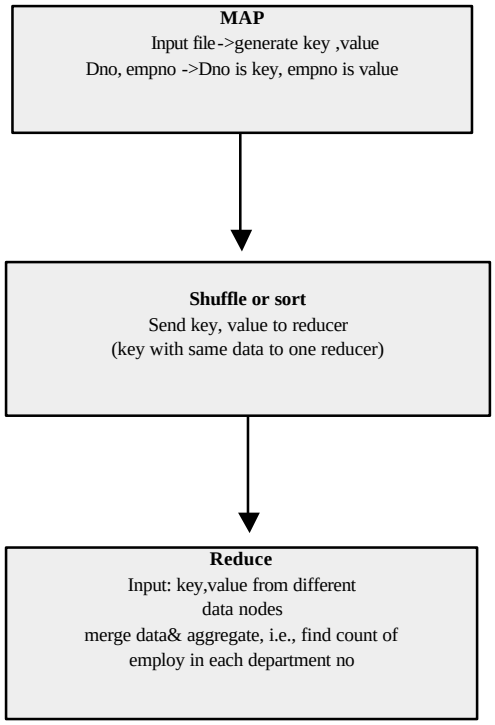

Fig. 1: Mapreduce flow for finding the number of employees in a department

MAPREDUCE PROGRAMMING OPTIONS IN HADOOP

Hadoop streaming: Streaming is a utility that comes with hadoop. It requires two executable files the mapper and the reducer. When a mapreduce job is submitted, map tasks are started in all datanodes. The input will be the stdin, so the input file is read line by line by the mapper (Fig. 2). The first tab character is used as the separator and before tab is considered as key and the rest the value. The output of the mapper is a collection of key and value written to stdout. Then, the reducer job is submitted which starts the reducer task in all datanodes. The reducer reads the keys and value from the stdin, line by line. It consolidates and writes out key and value to the stdout. The streaming command is given as from the bin directory hadoop home directory (White, 2012).

The hadoop jar~/hadoop-streaming.jar-input inputfile-output outputfile-mapper executable mapper-reducer executable reducer. Using streaming programs written in any language such as Java, Python, R and Ruby can be given in the mapper and reducer option.

Hadoop pipes: It is a programming interface for C/C++ programmers in hadoop. Map and reduce program has to

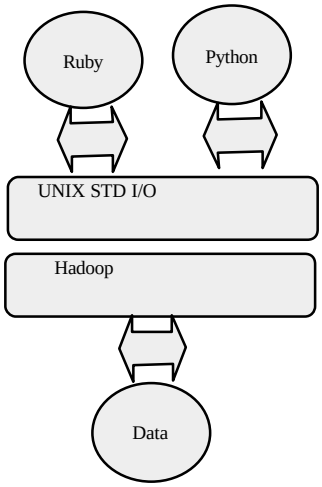

Fig. 2: Hadoop streaming

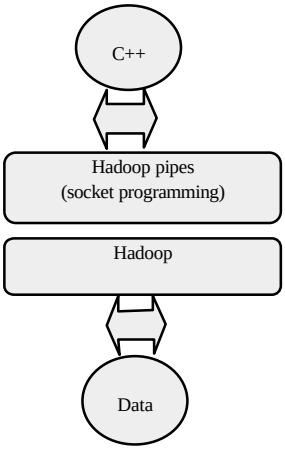

Fig. 3: Hadoop pipes

be written in C/C++. It works on the principle of the inter process communication channel, the sockets. The JVM creates and binds the server socket and puts in listen mode. The C/C++ wrapper starts the client socket and sends and receives messages to the server socket (Fig. 3). Map and reduce program read or write strings, represented using STL (Zhu *et al*., 2014). To run a program in hadoop the C/C++ source code, that contains the map and reduce function should be compiled using a makefile. Then, the object file and the data file are given in the hadoop command.

The hadoop pipes-D hadoop.pipes.java. record reader = true-D hadoop.pipes.java. recordwriter = true input path/file-output path-program path/filename.

Pig: Pig is a high level data flow interface for easily processing large data in hadoop (Gates *et al*., 2009). It makes the SQL programmers to taste the flavour of

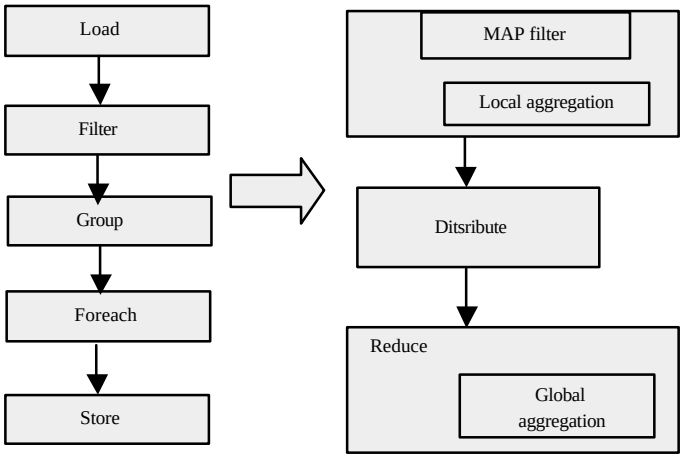

Fig. 4: Pig functions to mapreduce

mapreduce. Pig gives high level data manipulation constructs similar to SQL that are mapped to map, reduce functionalities. User gives what he want to retrieve in the form of sequence of operations expressed in pig which is considered as a logical plan, then converted to physical plan. The physical plan determines the map and reduce flow and then, the pig query is executed with mapreduce jobs. Parallel methods of executing relational operators (Raghu and Jojannes, 2003) are associated with each operation in the query. Example a "group by" query is executed in parallel by making local aggregation, global aggregation, distribution and merging. The above steps are mapped to a mapper and reducer which is shown in Fig. 4. The employee = LOAD 'data' USING BinStorage AS (SSN,name,DOB,address,Deptno); VE = FILTER employee BY address == 'Vellore'; GROUPS= group VE by Deptno NOE=FOREACH GROUPS GENERATE Deptno,count(*).

The above query , finds the number of employee who are in vellore for each deparment. To execute in parallel, i.e., using mapreduce, the mapper applies the filter, counts the number of employees in each department and sends the deptno and count to reducer based on the department number. That is each reducer receives count of one department or few more based on the total departments. Reducer then sums all the values of each department and provides the result.

HIVE: HIVE is a open source data warehouisng solution on top of hadoop, it supports a SQL like langauge called HiveQL (Thusoo *et al*., 2009). Operators in HiveQL are converted into mapreduce jobs. MapReduce scripts can also be embedded in HiveSQL. Data in HIVE is organized into tables, partitions and buckets (the terms follow the hierarchy). The HIVEQL provides DDL and DML statements like SQL. To name some, create, insert, select, project, join, aggregate, union all etc. A query in HIVEQL look like. FROM (SELECT e.no, e.dno) emp_dept select emp_dept.dno, count(1) GROUP BY emp_dept.dno. A Hive architecture as described in Fig. 5, consists of external interface-CLI /GUI / API in any language for user, thrift server-a framework for cross-language services, Metastore-system catalog, driver-life cycle manger of QL, the compiler = -converts QL into DAG of Mapreduce jobs, execution engine-hadoop to which mapreduce jobs are submitted.

JAQL: It is a Java Script Object Notation (JSON) query language, designed for JSON data and suits cloud environment. It is a functional query language where the queries are rewritten to use mapreduce. Jaql is widely used in IBM'S InfoSphere BigInsights and Cognos consumer insight products. It's deisgn has been influenced by pig, Hive and DryadLINQ (Beyer *et al*., 2011). It provides a higher level of abstraction to the user. Jaql comes with simple data model where a value is an array or atom or record. It supports no schema or partial schema specification and the schema is similar to XML Schema. Jaql scripts are sequence of statements flow diagram given in Fig. 6 that look like import myrecord; countemp = fn(records) (records-> transform myrecord::dno($) -> expand -> group by dno = $ as occurrences into { dept: dno, noe: count(occurrences)}); read(hdfs("emp.dat"))->countemp()->write(hdfs ("noe.dat")). The core expressions in Jaql are transform, expand, group by filter, join, union and tee.

Cascading: It is a thin java library for working with mapreduce operations on hadoop. The Java program contains SQL-like operators like group by Regular Expressions (REGEXP) and join.

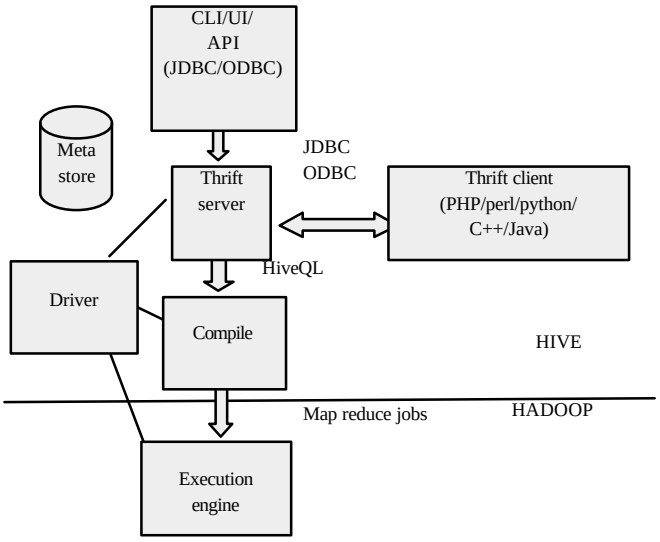

Fig. 5: HIVE architecture

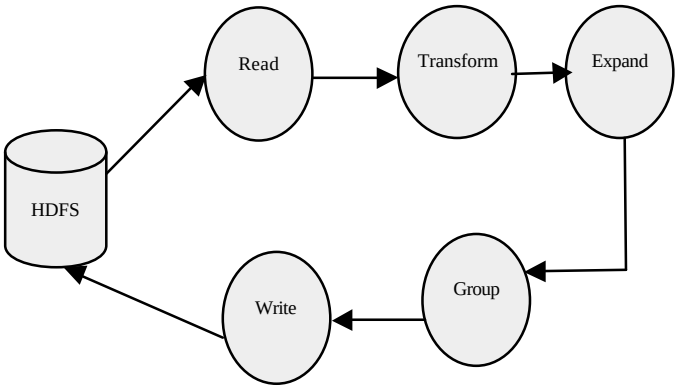

Fig. 6: Flow of Jaql query

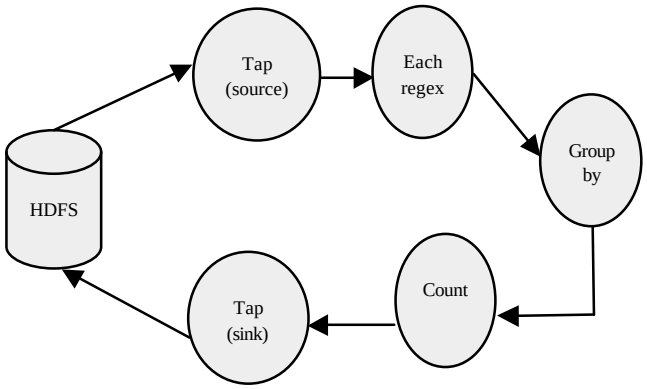

Fig. 7: Pipe assembly in cascading

Users can concentrate in writing what they want than how it should be executed in parallel. Cascading, the name comes from collection of flows where a flow is a pipe assembly binded to a tap. Pipe assemblies define what work should be done on a tuple stream. Tuple streams are read from tap source and wriiten to tap sink. Figure 7

shows the flow of a group by operation for a query like finding the number of employees in a department.

FUTURE ENHANCEMENT

Storing, indexing, querying and reasoning RDF data in cloud is vastly studied and researched (Kaoudi and Manolescu, 2015). Hence, the following two directions of research are recommended:

- To propose RDF storage methods in hadoop using any of the mapreduce options discussed in this study
- To propose methods for reasoning the RDF data stored in the mapreduce programming model using neighborhood systems (Tripathy *et al*., 2014)

CONCLUSION

The various methods for using mapreduce option in hadoop is discussed. Streaming and piping are options that allow the user to write their own mapreduce flow. Users have to concentrate more on mapreduce jobs than on queries where streaming allows java and other scripting languages (python, ruby), pipes are used for only c++ programs. Streaming is faster than pipes as it uses lower level I/O functions. The other approaches provide higher level of abstraction by hiding the mapreduce flows and concentrating on writing queries. Pig and Jaql are scripting languages with simple expressions for database operator. Hive provides a SQL like query language which is very adaptable for RDBMS users. In cascading, the database operators are written in Java.

REFERENCES

Beyer, K.S., V. Ercegovac, R. Gemulla, A. Balmin and M. Eltabakh et al., 2011. Jaql: A scripting language for large scale semistructured data analysis. Proceedings of the Conference on VLDB, August 29-September 3, 2011, IBM Research, Almaden San Jose, CA., USA., pp: 1272-1283.

Chandra, D.G., R. Prakash and S. Lamdharia, 2012. A study on cloud database. Proceeding of the 2012 4th International Conference on Computational Intelligence and Communication Networks (CICN), November 3-5, 2012, IEEE, New Delhi, India, ISBN:978-1-4673-2981-1, pp: 513-519.

Gates, A.F., O. Natkovich, S. Chopra, P. Kamath and S.M. Narayanamurthy et al., 2009. Building a high-level dataflow system on top of Map-Reduce: The Pig experience. Proc. VLDB. Endowment, 2: 1414-1425.

Kaoudi, Z. and I. Manolescu, 2015. RDF in the clouds: A survey. VLDB J., 24: 67-91.

Li, F., B.C. Ooi, M.T. Ozsu and S. Wu, 2014. Distributed data management using MapReduce. ACM. Comput. Surv., 46: 1-31.

Mladen, A.V., 2008. Cloud computing-issues, researches and implementations. J. Comput. Inf. Technol., 4: 235-246.

Ozsu, T. and P. Valduriez, 1991. Principles of Distributed Database Systems. Prentice-Hall, Englewood Cliffs, NJ., USA., ISBN-13: 9780136916437, Pages: 562.

Raghu, R. and G. Jojannes, 2003. Database Management Systems. Mc-Graw-Hill, New York, USA., ISBN:0072465638 9780072465631,.

Sakr, S., A. Liu and A.G. Fayoumi, 2013. The family of map reduce and large-scale data processing systems. ACM. Comput. Surv., 46: 1-11.

Taniar, D., C.H. Leung, W. Rahayu and S. Goel, 2008. High Performance Parallel Database Processing and Grid Databases. Vol. 67, John Wiley & Sons, Hoboken, New Jersey, ISBN:978-0-470-10762-1, Pages: 553.

Thusoo, A., J.S. Sarma, N. Jain, Z. Shao and P. Chakka et al., 2009. Hive: A warehousing solution over a map-reduce framework. Proc. VLDB. Endowment, 2: 1626-1629.

Tripathy, B.K., H.R. Vishwakarma and D.P. Kothari, 2014. Neighbourhood systems based knowledge acquisition using MapReduce from Big Data over cloud computing. Proceeding of the (CSIBIG), 2014 Conference on IT in Business, Industry and Government, March 8-9, 2014, IEEE, Tamil Nadu, India, ISBN:978-1-4799-3064-7, pp: 1-6.

White, T., 2012. Hadoop: The Definitive Guide. O'Reilly Media Inc., Sebastopol, California, ISBN: 978-1-449-31152-0, Pages: 659.

Zhu, J., J. Li, E. Hardesty, H. Jiang and K.C. Li, 2014. GPU-in-Hadoop: Enabling MapReduce across distributed heterogeneous platforms. Proceeding of the 2014 IEEE-ACIS 13th International Conference on Computer and Information Science (ICIS), June 4-6, 2014, IEEE, Arkansas, USA., ISBN:978-1-4799-4860-4, pp: 321-326.