International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 03 | Mar-2016 www.irjet.net p-ISSN: 2395-0072

Bidirectional Hadoop Kakfa Managing Messaging Bus

Prachi M. Birajdar [1], Kanchan Ujede [2], Rohini Yalawar [3], Kailas H. Biradar[4] , Zeeshan Khan [5], Swapnil Chaudhari [6]

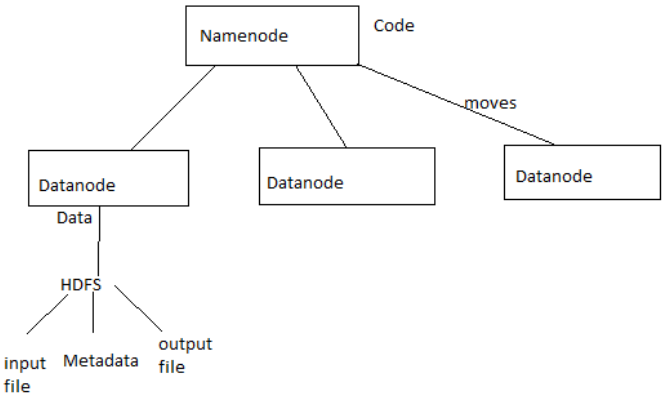
[1234] Student, Computer Engineering Department, MMIT, Maharashtra, INDIA
[5] Cloud Automation Java Developer , Maharashtra, INDIA
[6] Professor, Computer Engineering Department, MMIT, Maharashtra, INDIA

Abstract - In this paper, We introduce Kafka, a robust messaging system that we developed for collecting and delivering high volumes of data. Our system incorporates ideas from existing messaging systems, and is suitable for message consumption. Using hadoop framework which stores large amount of unstructured data. To develop bidirectional communication via kafka connector ,the data can be send without any data loss. Our expected results will produce superior performance of kafka connector when compared with two popular messaging system.

Key Words: *Key word1, Hadoop , Kafka , zookeeper*

1. INTRODUCTION

Hadoop is a combination of two domain that is distributed system and bigdata. Hadoop has 100% guarantee to secure of data and availability. In hadoop echo system data at one place and code will move to the data.



1.1. HDFS

The Hadoop Distributed File System is based on the Google File System and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It has to be low cost hardware and fault tolerant. It provides high throughput access to application data and is suitable for applications having large datasets.

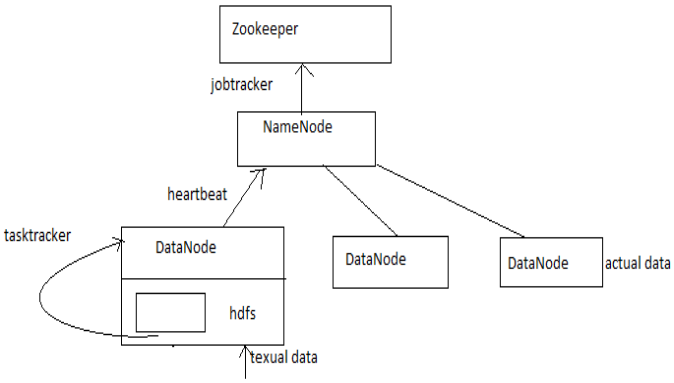


fig. HDFS Architecture

Fig-1 : HDFS architecture

1.2 Zookeeper

Zookeeper is a thread and also demon process. DataNode:actual work done by it.

NameNode If namenode will dead then it will give task on one of the datanode which is nearer to that namenode.

1.3 Hbase

It is combination of hadoop distributed file system. It is used to store all the information about entire the echo system.

2. KAFKA

Kafka is a managing messaging bus. In this kafka, partition is takes place. partition consist of number of topics. Each partition has a particular group id that is groupid1 are in two partition and by default kafka is empty. It only includes partition ,partition is done on physical memory.

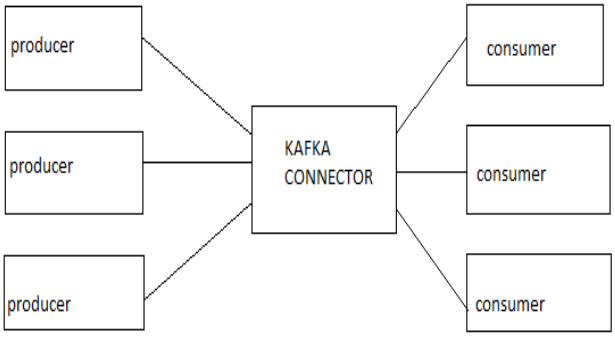


Fig-2: kafka overview

2.1. MESSAGING TERMINOLOGY

Kafka ; Maintains number of messages in categories called *topics.* We will call processes that publish messages to a Kafka topic *producers.* We will call processes that subscribe to topics and process the number of published messages *consumers.* Kafka is run as a cluster. it is comprised of one or more servers each of which is called a *broker*.

Producer : Producer produce data to the topics on their own choice. The producer is responsible for choosing its own messages and it is assign to which partition within the topic and this is depends on producer.

Consumer : Consumers has its own consumer group name, and each message published to a topic. It is delivered to one consumer within each subscribing consumer. When all the consumer instances have different consumer groups, then it works like publish-subscribe. all messages are broadcast to all consumers.

2.2. ALGORITHM

Step 1 : Producer will produce data.

Step 2 : Data will be stored in kafka bus.

Step 3 : Consumer will consume the data.

Step 4 : Consumer will produce data.

Step 5 : Data will stored in kafka bus .

Step 6 : Producer will consume the data.

2.3. WORKING

The bidirectional kafka is nothing but the messaging bus for transferring the data,which is produced by the producer and consume by the consumer and vice versa through kafka connector

There are three modes of messaging system that are:

1. Write only (Producer)
2. Bidirectional
3. Sinker(Consumer)

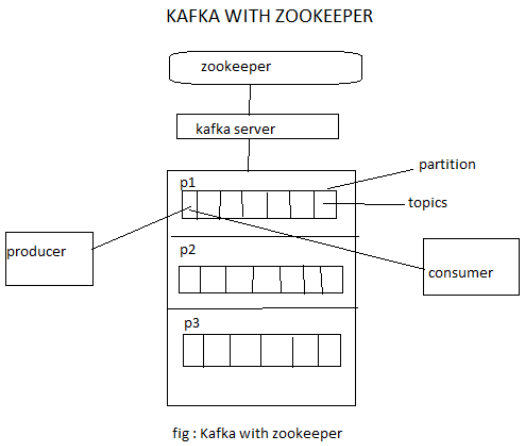


fig : Kafka with zookeeper

Fig -3: Kafka with zookeeper

Kafka server is act as middleware which can send the details of topics and partition to the Zookeeper server.

Zookeeper server is a thread and also demon process,

highly available that can monitor and co-ordinate kafka server as well as overall system details. zookeeper detecting addition and removal of kafka bus and consumers.
Kafka buffer contains multiple topics and partitions, Over all details of topics and partitions cannot directly monitored by zookeeper, kafka server contains all the details then kafka server communicate with zookeeper.

A topic is a category, where messages are stored. For each topic, the Kafka bus contains a partition. One partition contains one or more than one topics.

If more than one producer wants to produce the data at same time they have to wait, till one producer produce the data into topic, after than other producer can produce the data into topic. This will causes the delay. To avoid that delay kafka bus allows multiple producer to produce their data into separate topics and that related topics combined together into one partition that have assigned a group id. If some related topics is in different partitions then consumer can consumes the related topics that have assigned same group ID and consumes data

Kafka bus contains all published data, that data can store for some configurable time period which is consume by consumer
Consumer consume the data from the topics. whatever the data producer produces into topic that is received by consumer. There is a list of topic into consumer side which can helps the consumer to consume the data as per his choice

Producer produce data to the topic and consumer consume the data form topics, this is the flow of unidirectional kafka connector.

It is also possible bidirectional that consumer produces the data and producer consumes the data.

Consumer act as a producer that accept the data from end user and produce to topic which can be stored into partition that have assigned some group ID numbers which can be helpful to consume the data at producer side. Using group ID and the topic name, producer can receives data.

all system details means overall details about producer, consumer, kafka server is monitored by the zookeeper.

All the modules are consistently giving acknowledgement to the zookeeper.

If zookeeper doesn't receive any acknowledgement from the modules then it realies that module is dead or getting down.

Produced data is delivered to consumer is known to producer by getting acknowledgement from zookeeper because consumer constantly communicate with zookeeper and zookeeper then sends acknowledgement to producer.

Producer is sending data and at that time consumer gets down then the data is stored into kafka bus. Whenever consumer gets up he received that data, hence there is no loss of data.

When producer is producing some data to kafka bus and suddenly disconnects from kafka bus then remaining data will be produced after producer gets up, it is not necessary to send whole data. Hence duplication of data is avoided.

3. CONCLUSIONS

In this paper, we have presented a system to pass the messages bidirectionally using kafka buffer.huge amount of data can be send from producer to consumer and vice versa. Moreover flexibility is increased.This work is the first step to contribute to the real time implementation of the system.The early encouraging results we obtained only aimed to improve the quality of messaging system.Our present work is limited to the text messages based only.In particular future plans are to contribute to different languages. Additionally we plan to desire a more sophisticated approach to when a user should be penetrated into a black lists.

REFERENCES

[1] J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," In Proc. of the 6th Symposium on Operating Systems Design and Implementation, San Francisco CA, Dec. 2004.

[2] A. Gates, O. Natkovich, S. Chopra, P. Kamath, S. Narayanam, C. Olston, B. Reed, S. Srinivasan, U. Srivastava. "Building a High-Level Dataflow System on top of MapReduce: The Pig Experience," In Proc. of Very Large Data Bases, vol 2 no. 2, 2009, pp. 1414–1425

[3] H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur. "PVFS: A parallel file system for Linux clusters," in Proc. of 4th Annual Linux Showcase and Conference, 2000, pp. 317–327.

[4] F. P. Junqueira, B. C. Reed. "The life and times of a zookeeper," In Proc. of the 28th ACM Symposium on Principles of Distributed Computing, Calgary, AB, Canada, August 10–12, 2009.

[5] K. V. Shvachko, "HDFS Scalability: The limits to growth," ;login:. April 2010, pp. 6–16.

[6] T. White, Hadoop: The Definitive Guide. O'Reilly Media, Yahoo! Press, June 5, 2009.

[7] F. P. Junqueira, B. C. Reed. "The life and times of a zookeeper," In Proc. of the 28th ACM Symposium on Principles of Distributed Computing, Calgary, AB, Canada, August 10–12, 2009.

[8] Apache Hadoop. http://hadoop.apache.org/

[9] Hadoop File System http://hadoop.apache.org/hdfs/

[10] Zookeeper http://hadoop.apache.org/zookeeper/

[11] Kafka http://sna-projects.com/kafka/

[12] JAVA Message Service: http://download.oracle.com/javaee/1.3/jms/tutorial/1_3_1fcs/doc/jms_tutorialTOC.html.