www.ioirp.com
International Journal of Innovative Research in Technology, Science & Engineering (IJIRTSE)
ISSN: 2395-5619, Volume – 1, Issue – 3. May 2015

IMAGE ANALYTICS ON BIG DATA IN MOTION – IMPLEMENTATION OF IMAGE ANALYTICS CCL IN APACHE KAFKA AND STORM

Lokesh Babu Rao[1], Prof.C.Elayaraja[2]
[1] PG Student, Department of ECE, Dhaanish Ahmed College of Engg, TamilNadu, India
[2] Head of Department, M.E. Applied Electronics, Dhaanish Ahmed College of Engg, TamilNadu, India

***Abstract*— Continuous stream of Image and Video data processing is a big data challenge. Though the amount of data is comparatively small, it requires special resources to process the data since it is in continuous motion. Efficient transfer of Video / Image data to the centralized Data processing location is also a potential problem. Implementing Connected Component Labelling (CCL) using Apache Kafka and Apache Storm solves the problem of efficient data transfer and Centralized Image processing. Apache Kafka is an Open Source messaging system used for large data transfer across several data sources via a centralized cluster. Apache Storm is a Distributed Processing System used to process big data on a cluster.**

Key Words: Big Data, Apache Kafka, Apache Storm, Connected Component Labelling (CCL)

I. INTRODUCTION

Big data can be best defined as the data which has Volume, Variety and Velocity. Big data on image analytics is best applicable on processing photos and Surveillance videos. Processing Surveillance videos for information extraction requires real time stream processing. The Video data will be continuous in motionwhich can reach massive proportions over time. The Video data requires to get processed on time to extract the full benefit of surveillance.About 2.5 quintillion bytes of data is getting generated every day and 90% of all data was generated after 2011. Generally, big data is defined based on its main characteristics which are growing in three dimensions: volume, velocity and variety. In this new concept, the volume of unstructured data is in the scale of petabytes, and creation of them is in the fraction of the second. In the literature big data in motion is defined as continues data streams at high data transfer rate. Such big data represent data sets that cannot be analyzed with conventional algorithms and standard hardware platforms. Due to the typical memory capacity and bandwidth limitations, which determine the overall throughput, the processing of the continuously increasing amount of data is done online and locally on the streamed data. The new scale of volume, velocity and variety requires redesigning a number of infrastructure components and algorithms to support the large-volume, complex and growing data.

Scientific data are defined in four different types: raw data, structured data, published data and data linked to the publication. First type of the scientific data is raw data which is generated from observation and experiment of different phenomena. For instance biological, climate and life sciences generate massive amounts of scientific data. Images and videos have the highest amount of volume among scientific data, and are analytically prepared to gain additional value. The preparation is done by extracting various properties of the image such as objects and movements.
The processing of these large image collections usually requires large amounts of computational power. In addition to the number of images, there are other factors that influence the computational complexity of the processing and analysis tasks: the increasing resolution of images (e.g. mega images and experimental workflows that involve repeated algorithm execution varying different parameters. A solution to this problem is to use computational infrastructures (such as clusters and supercomputers) that can cope with the task. Stream processing clusters with the ability to collect the Data Stream efficiently is the need of the hour for the scientific community

II. BIG DATA PROCESSING

Image processing requires Stream data processing for the following reasons: 1.Video processing requires on time processing for surveillance 2. Processing video data after some span of time is hard and memory intensive. In order to process Image data, Data streaming or live data transfer from source to the Stream data processing cluster has to be dealt efficiently. Apache Kafka deals the problem of live data transfer with high efficiency.

A. Apache Kafka

Apache Kafka is an open-source message broker project developed by the Apache Software Foundation written in Scala. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. The design is heavily influenced by transaction logs.

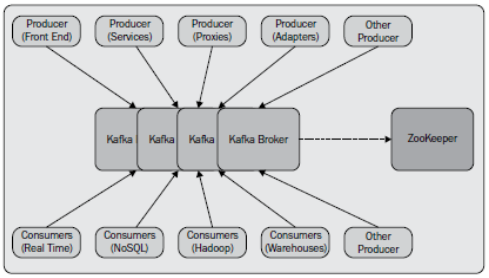

Fig 1: Kafka-Architecture

To derive the real value from big data, any kind of information loss cannot be afforded. Apache Kafka is designed with O (1) disk structures that provide constant-time performance even with very large volumes of stored messages, which is in order of TB.

B. Apache Storm

Apache Storm is a distributed real-time computation system for processing fast, large streams of data. Storm adds reliable real-time data processing capabilities to Apache Hadoop® 2.x. Storm in Hadoop helps capture new business opportunities with low-latency dashboards, security alerts, and operational enhancements integrated with other applications running in their Hadoop cluster.

Fig 2: Storm-overview

Storm is a distributed, reliable, fault-tolerant system for processing streams of data. The work is delegated to different types of components that are each responsible for a simple specific processing task. The input stream of a Storm cluster is handled by a component called a spout. The spout passes the data to a component called a bolt, which transforms it in some way. A bolt either persists the data in some sort of storage, or passes it to some other bolt. You can imagine a Storm cluster as a chain of bolt components that each makes some kind of transformation on the data exposed by the spout.

III. CONNECTED COMPONENT LABELLING (CCL)

To demonstrate the applicability of Image analytics in Big Data Stream processing CCL Algorithm is implemented. Connected component labeling (alternatively connected-component analysis, blob extraction, region labeling, blob discovery, or region extraction) is an algorithmic application of graph theory, where subsets of connected components are uniquely labeled based on a given heuristic. Connected-component labeling is not to be confused with segmentation. Connected-

component labeling is used in computer vision to detect connected regions in binary digital images, although color images and data with higher dimensionality can also be processed.

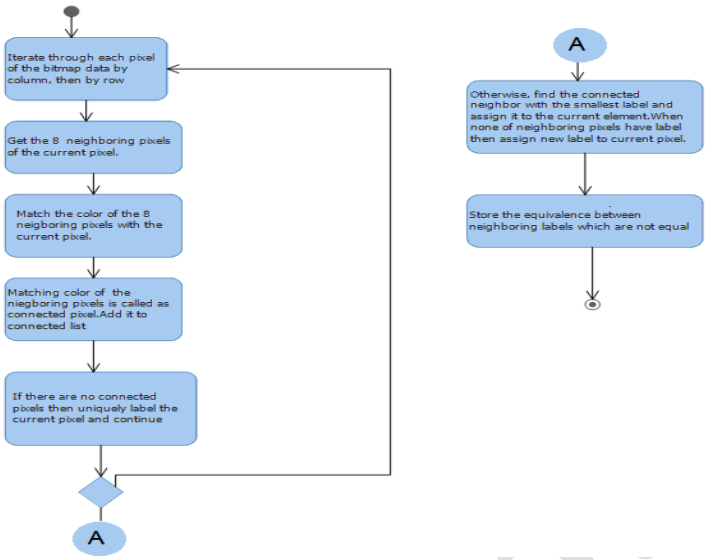

Fig 3:CCL-Algorithm

Fig-2 represents the basic CCL Algorithm. Connected components labeling scans an image and groups its pixels into components based on pixel connectivity, *i.e.* all pixels in a connected component share similar pixel intensity values and are in some way connected with each other. Once all groups have been determined, each pixel is labeled with a gray level or a color (color labeling) according to the component it was assigned to. Extracting and labeling of various disjoint and connected components in an image is central to many automated image analysis applications.

IV. CCL IMPLEMENTATION

The CCL Image analytics is implemented in Storm Topology in CCL Bolts. Topology is the graphical representation of Stream Data processing with Bolts as its nodes. Bolts are the processing units of Storm which acts as Virtual Task agent. The Kafka Agent code reads the Image from the location specified and converts it into String. The Kafka Agent can collect and stream data from multiple sources. The Collected Image string is then emitted to the Kafka cluster under specific Kafka Topic. Topic Name is the unique Stream Id applied to the Kafka Cluster by setting the configurations. The Kafka cluster receives the message and stores it in the cluster under the specified topic name.

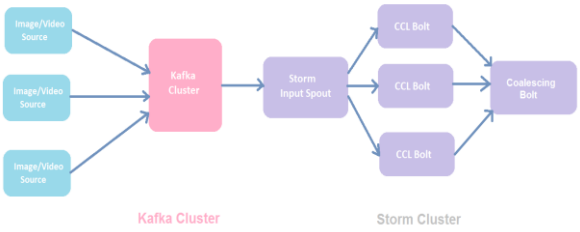

Fig 4:CCL-Topology

The topology in Figure: 4contains the Kafka Spout which listens to the specified Kafka Topic name. When the message is received into the Kafka Cluster, Kafka Spout receives the copy of the message. The Kafka Spout then transmits the data to the CCL Bolt connected to it. The CCL bolt is implemented using the CCL algorithm. The image is reconstructed in the bolt from String to PNG format. The PNG file is converted into Portable Pitmap format (PPM) using Image Magic Tool.

The CCL Algorithm is applied to the PPM file where multiple loops over the data is applied to get the final PPM output. The output PPM file is saved in the specified directory.Thus, the image is efficiently transmitted using Apache Kafka and Processed live using Apache Storm.

V. IMPLEMENTATION RESULTS

The CCL-Topology is tested using Kafka-Storm single node Cluster. It is tested using 300Kb Image dataset with standard CCL Algorithm. The processing time (including the Kafka transfer time) is measured for performance calculation.

Number of Tasks/Cores	4	8	16
Processing Time (Seconds)	16.0567	10.8952	5.5432

Table 1: Number of Cores Vs Processing Time

The above table denotes the processing time of 300Kb image dataset for different number of Cores / Tasks.

A. Comparing Kafka-Storm with other systems

The Kafka-Storm system is compared with the following existing systems, (1) BIGS: A Framework for Large-Scale Image Processing and Analysis over Distributed and Heterogeneous Computing Resources, (2) CUDA: CUDA-enabled Hadoop Cluster for Fast Distributed Image Processing.

BIGS

Big Image Data Analysis Toolkit is a software framework for large scale image processing and analysis over heterogeneous computing resources, such as those available in clouds, grids, computer clusters or throughout scattered computer resources (desktops, labs) in an opportunistic manner. Through BIGS, eScience for image processing and analysis is conceived to exploit coarse grained parallelism based on data partitioning and parameter sweeps, avoiding the need of inter-process communication and, therefore, enabling loosely coupled computing nodes.

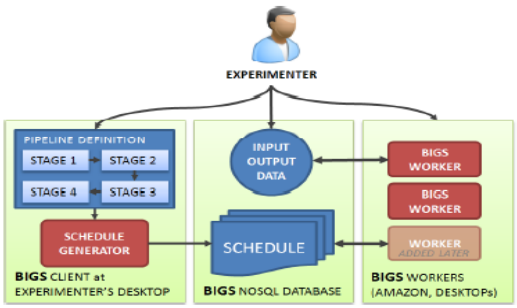

Fig 5: BIGS Overview

BIGS adopts an uncommitted resource allocation model where experimenters define their image processing pipelines in a simple configuration file, a schedule of jobs is generated and workers, as they become available, take over pending jobs as long as their dependency on other jobs is fulfilled. BIGS workers act autonomously, querying the job schedule to determine which one to take over. This removes the need for a central scheduling node, requiring only access by all workers to a shared information source.

5.1.2 CUDA

CUDA or Compute Unified Device Architecture is a C-based programming model proposed by NVIDIA for leveraging the parallel computing capabilities of the GPU for general purpose computations.

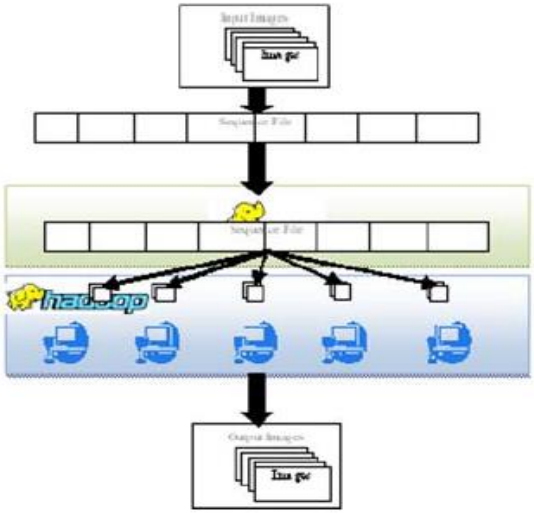

Fig 6: CUDA – Hadoop Processing

They are attempting to integrate CUDA acceleration into the Hadoop distributed processing framework to create a heterogeneous high performance image processing system.Hadoop is a map-reduce based distributed processing framework, frequently used in the industry today, in areas of big data analysis, particularly text analysis. Graphics processing units (GPUs), on the other hand, are massively parallel platforms with attractive performance to price and power ratios, used extensively in the recent years for acceleration of data parallel computations. The Hadoop Architecture for small files is shown in Figure 6.

KAFKA-STORM COMPARISION

The approximate processing time of BIGS & CUDA systems are obtained for different datasets. The 300Kb dataset performance is compared with Kafka-Storm System.

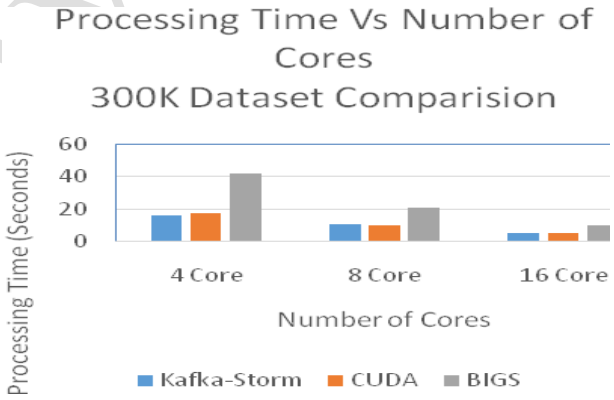

Fig 7:Comparison plot: Kafka-Storm Vs Other Systems

The above bar chart represents the Processing Time of Kafka-Storm, CUDA & BIGS for 300Kb data set for multiple Core / tasks. From the above comparison, the Kafka-Storm system's performance is similar to CUDA system and better than BIGS system.

VI. FUTURE WORKS

The Kafka-Storm Topology is being designed to analyze Surveillance video. The system will extract Human faces from Surveillance video and compares with faces in Database to detect. The system is a real time face recognition for Surveillance video.

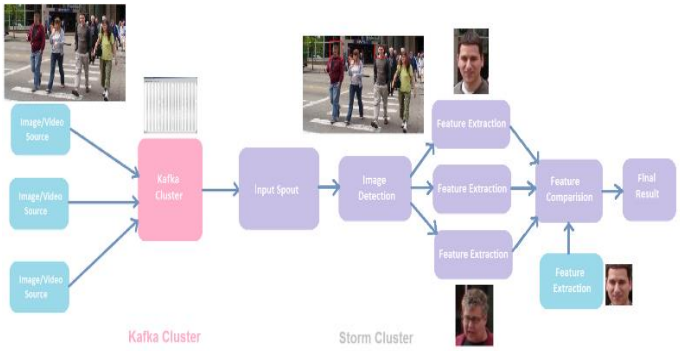

Fig 8: Real-Time Face Recognition using Kafka-Storm

The Kafka – Agent collects the images from various videos sources at a desired frequency. It transmits the image string data to the Kafka Cluster. The entry level of the Storm Topology is the Spout which receives the String data from Kafka Cluster.
The Image String data is reconstructed in to png image and transferred to Image Detection Bolt. Image Detection Bolt is implemented using OpenCV library which uses Principal Component Analysis using Eigen faces. The Faces are separated in this bolt and applied to the Face Recognition bolt. The Face Recognition Bolt is connected to a Database which contains the images to be recognized and its feature values. The Face Recognition Bolt compares the current faces with the Database faces using Feature values. The faces which are matched are transferred to the final result bolt.

VII. CONCLUSION

The CCL Algorithm is successfully implemented in Apache Kafka and Apache Storm. The performance comparison results show that the Kafka-Storm System is similar to CUDA system and better than BIGS System. In addition to Image stream processing, Kafka –Storm System provides Image Stream data transfer from multiple image sources.Thus Kafka-Storm system provides complete solution for Big-data Image analytics.

References

[1] S.M. Najmabadi, M. Klaiber, Z. Wang, Y. Baroud and S. Simon Stream "*Processing of Scientific Big Data on Heterogeneous Platforms – Image Analytics on Big Data in Motion*", Institute for Parallel and Distributed Systems.

[2] U. Chandrasekhar, A. Reddy, and R. Rath, "*A comparative study of Enterprise and open source big data analytical tools*," Conference onInformation Communication Technologies (ICT), 2013 IEEE, pp.372–377.

[3] T. Eaton, D Deroos, "*Understanding big data*," in McGraw-Hill Companies, April 2012.

[4] Raúl Ramos-Pollán, Fabio A. González, Juan C. Caicedo, Angel Cruz-Roa, Jorge E. Camargo, Jorge A. Vanegas, Santiago A. Pérez, Jose David Bermeo, Juan Sebastián Otálora, Paola K. Rozo, John E. Arévalo -"*BIGS: A Framework for Large-Scale Image Processing and Analysis Over Distributed and Heterogeneous Computing Resources*"

[5] Ranajoy Malakar & Naga Vydyanathan - "A CUDA-enabled Hadoop Cluster for Fast Distributed Image Processing".

[6] https://kafka.apache.org

[7] https://storm.apache.org