IMAGE CLOUD PROCESSING FOR BIGDATA USING STATIC IMAGE CLOUD PROCESSING WITH HADOOP

Suvathi. D [1] Dr. Ruba Soundar. K [2]

ABSTRACT

Large quantities of images are shared through multi platform services. The most image cloud processing applications, designed for large scale image. The Hadoop framework which provides a hadoop based library to support large scale image. The framework provides a high level transparency and efficiency for creating large scale image processing applications on top of hadoops MapReduce framework. The proposed Image Cloud Processing (ICP) framework consists of two mechanisms. Static ICP (SICP), Dynamic ICP (DICP). SICP and DICP representation named P- Image and B- Image are designed to cooperate with MapReduce. SICP is processed mainly for efficiency processing large scale images that have already been stored in the distributed system. DICP implemented through a parallel processing procedure working with the traditional processing mechanism of distributed system.

Key Words: Distributed System, Image Processing, Bid data, Cloud Computing

1. Introduction

Big data is sprouting up everywhere and using it appropriately will drive competitive advantage. Ignoring big data will put on organization at risk and cause it to fall behind the competition. To stay competitive, it is imperative that organizations aggressively pursue capturing and analyzing these new data source to gain the insights they offers. Big data is not about the size of the data in terms of how much data there is. According the group the “big” in big data also refers to several other characteristics of a big data sources. While big data certainly involves a lot of data does not refers to data volume alone. Big data also has increased velocity, complexity and variety of data source of the past. It is needless to say how important of image classification/ recognition is in the field of computer vision – image recognition is essential for bridging the huge\ semantic gap between an image, which is simply a scatter of pixels to untrained computers, and the object it presents. Therefore, there have been extensive research efforts on developing effective visual object recognizers. Most research efforts on image classification so far have been focused on medium-scale datasets, which are often defined as datasets that can fit into the memory of a desktop. We then cluster graph vertices associated with the compatibility matrix and extract its dominant set as the optimal matches.

2. Related Work

Image cloud processing is implemented in parallel and wants a framework structure for image processing and gain a raise in time efficiency without compromising the result. SICP representation named P- Image and B- Image are designed to cooperate with MapReduce [1]. There are two main reasons for the limited effort on large-scale image classification. First, until the emergence of Image Net dataset, there was almost no publicly available large-scale benchmark data for image classification. This is mostly because class labels are expensive to obtain. Second, large-scale classification is hard because it poses more challenges

[1] Author, M.E Scholar, Department of Computer Science and Engineering, P.S.R. Engineering College, Sivakasi, India,

[2] Co –Author, Professor, Department of Computer Science and Engineering, P.S.R. Engineering College, Sivakasi, India,

than its medium-scale counterparts [5]. A key challenge is how to achieve efficiency in both feature extraction and classifier training without compromising performance. Cloud computing is an emerging commercial infrastructure paradigm that promises to eliminate the need for maintaining expensive computing facilities by companies and institutes alike. Thus, clouds have the potential to provide to their owners the benefits of an economy of scale and, at the same time, become an alternative for scientists to clusters, grids, and parallel production environments [4]. Hadoop Distributed File System (HDFS) is widely used in large-scale data storage and processing. HDFS uses MapReduce programming model for parallel processing. The work presented in this paper proposes a novel Hadoop plug-in to process image files with MapReduce model. [9]The proposed technique is based on merging multiple small size files into one large file to prevent the performance loss stemming from working with large number of small size files. In that way, each task becomes capable of processing multiple images in a single run cycle. The effectiveness of the proposed technique is proven by an application scenario for face detection on distributed image files. *MapReduce* framework to implement our algorithms and demonstrated the performance in terms of classification accuracy, speedup and scale up using a wide variety of synthetic and real-world data sets. Building the next generation of multimedia systems for content retrieval navigation and browsing hinges on solving critical tasks such as image classification, event detection, and video summarization. In turn, these tasks are based on accurately detecting semantic categories from very large repositories of visual information, i.e., images and videos. In practical applications, this key classification problem is further compounded by the visual ambiguity induced by a noisy content. Although a lot of research has been conducted to address semantic classification of visual information, most techniques from the literature hardly address the fundamental problem of blurry and noisy content. However, low-quality content ranging from noisy to motion-blurred key frames is prevalent in broadcasting applications and even more in currently pervasive nonprofessional user-generated images and videos. The underlying approach is based on three strategies: extraction of essential signatures captured from a global context, simulating the global pathway; highlight detection based on local conspicuous features of the reconstructed image, simulating the local pathway; and hierarchical classification of extracted features using probabilistic techniques.

3. Static Image Cloud Processing

Our ICP framework consists of two complementary processing mechanisms, i.e., SICP (Static Image Cloud Processing) and DICP (Dynamic Image Cloud Processing). As shown in Fig.5.1, SICP is aimed at processing those large-scale image data that have been stored in the distributed system. Decode these static images first to maintain the necessary information as their corresponding P-Images which will be then stored in the data file contained in Big- Image. Then, when image processing is required, we just need to index the index file also stored in Big-Image to find the demanded P-Images which provide the needed image information. Traditional image processing methods usually utilize the small image files as serial processing units, which seriously limits the processing efficiency and even results in a breakdown once the cluster fails to timely process such huge amounts of small image files. To overcome these constraints, our SICP mechanism provides a different design. The dataset contains 30 image groups, each of which represents a distinct scene or object. The first image of each group is the query image and the correct retrieval results are the other images of the group.

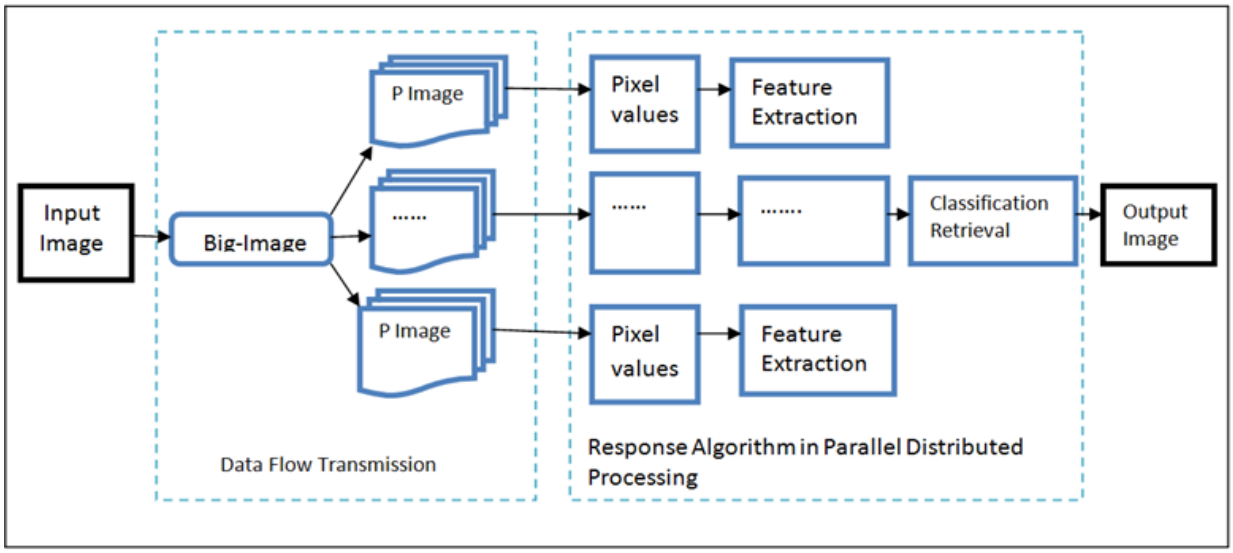

Fig 3.1System architecture diagram

3.1 Modeling for P- Image

P-Image includes the filename, the pixel values, and the width height of the initial image. To our best knowledge, almost all of the image processing algorithms in computer vision are based on pixel information. Once contained in P-Image, this information would not get lost and hence, time consumption will be greatly reduced by avoiding the repeated decoding operations. In our design, we utilize a two-dimensional matrix to store the pixel values corresponding to those stored in the P-Image. By accessing the matrix, we can obtain the pixel values at a high speed owing to the one-to-one correspondence between the pixel coordinates recorded in the matrix and those contained in P-Image.

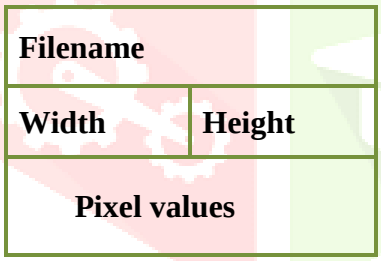

Fig 3.2 Structure of P-Image

The structure of Big-Image which consists of a data file and an index file. The data file is employed to store the aforementioned P-Images, and the index file is utilized to record the ID and Offset of each P-Image stored in the data file. Here, we store the P-Images in Big-Image so as to save memory space, avoid a loss of image information, and process huge amount of images at a time. The catalogue of the index file is made up of two fields, i.e. ID and Offset. The P-Image ID is computed by the Hash function with the P-Image filename, and the P-Image Offset denotes its corresponding location in the data file. Indexing through the index file using the ID to get the corresponding offset, we can directly get the P-Images stored in the data file to extract the needed image information for subsequent processing. Compared with the traditional small image files, Big-Image effectively avoids the queuing delay.

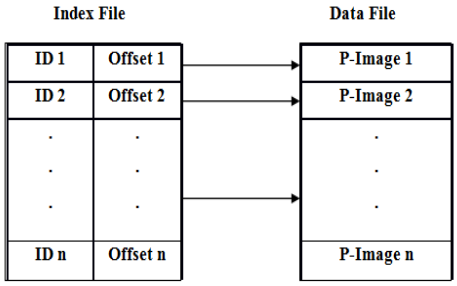

Fig 3.3 Structure of B-Image

In image processing field, RGB and Grey color modes are the most widely used color space to represent images. As depicted in RGB color mode contains a 1*3 array to store the key of the three channels. By contrast, Grey color mode only contains a single key. Despite this difference, the key ranges from 0 to 255 no matter what the color mode is. Sometimes, we need to transform the RGB mode to Grey mode when using P-Image. In our work, we employ a famous formula of psychology to accomplish the transformation:

M(x; y) =M(x, y) R*0.2989 +M(x,y)G * 0.5870+M(x,y)B*0.1140 →(1)

Partition of Big-Image, each Map Node would deal with its corresponding GP[k] (the k^{th} group of P-Images) in parallel to gain pixel values and accomplish feature extraction. We call this processing procedure mapping function, i.e. M {.}, which takes GP[k] as input. The eventually gained features can be defined as

FI(k)=M{GP[k]} →(2)

Where FI (k) represents the total features of each GP[k] after the M {.} operation. The gained FI (k) would act as input to the reducing function, i.e. R {.}, in which αk is another input coefficient (we will define the role of αk shortly). The final output θ is expressed as equation (3)

$\theta=R\{\sum_{k=1}^{num\ mapTask}(\alpha k)F1(k)\}$ **→(3)**

6.1.2 Mapping a Image Dataset

Framework also designed for improving computational efficiency and with a strong focus on real applications. The image processing, P-Image producing, and Big-Image producing can be implemented in parallel, which demonstrates that producing P-Image and Big-Image will not drag down the whole efficiency.

File Name	Width	Height	Size
1.jpeg	1024	768	108780
2.jpeg	1024	768	142560
3.jpeg	1024	768	64671
4.jpeg	1024	768	168427
5.jpeg	1024	768	131269
6.jpeg	1024	768	69429
7.jpeg	1024	768	81983
8.jpeg	1024	768	81983
9.jpeg	1024	768	104265
10.jpeg	1024	768	86857

File Name	Width	Height	Size
11.jpeg	1024	768	119955
12.jpeg	1024	768	93106
13.jpeg	1024	768	170695
14.jpeg	1024	768	170695
15.jpeg	1024	768	116026
16.jpeg	1024	768	90724
17.jpeg	1024	768	206963
18.jpeg	1024	768	175553
19.jpeg	1024	768	162535
20.jpeg	1600	1200	421189

File Name	Width	Height	Size
21.jpeg	1024	768	135616
22.jpeg	1024	768	127504
23.jpeg	1024	768	149564
24.jpeg	1024	768	115924
25.jpeg	1024	768	57099
26.jpeg	576	768	136072
27.jpeg	576	768	92376
28.jpeg	576	768	58960
29.jpeg	576	768	64438
30.jpeg	576	768	64438

Table 6.1 Splitting an image based on File Name, Width, Height and Size

4. Feature Extraction

All of the operations, that are partitioning Big-Image, gaining pixel values, extracting features, mapping, reducing, etc., can be implemented in parallel. Apparently, compared with traditional methods taking single image file as input to be processed serially, P-Image and Big-Image contribute a lot to the parallel processing in SICP mechanism. Instead of suffering a breakdown of the cluster, SICP guarantees a stable processing procedure even when the image scale reaches a huge extent, which largely attributes to the high scalability that the cloud computing platform owns. With the excellent cloud computing capability, the

image processing equals to processing several big image files after the simple partition of Big-Image. Big Image data Processing on SICP,

Step 1: Produce P-Images

P-Image: an image representation containing filename, width-height, and pixel values;
Input: Image [1 : : : N];

1: for each Image[i] do
2: Decode Image[i] to retain its filename, width, height, and pixel values;
3: end for

Step 2: Produce Big-Image

Big-Image: a huge file containing a data file and an index file;
ID: image filename; Offset: P-Image[i]'s size; Offset 0;

4: for each P-Image[i] do
5: Offset ← Offset + the size of P-Image[i];
6: Insert ID and Offset of P-Image[i] into Big-Image.index;
7: Insert P-Image[i] into Big-Image.data;
8: end for

Step 3: Partition Big Image

GP[k]: the kth group of P-Images;
BLOCKSIZE: the size of GP[k];
NumMapTask: the number of Map Nodes;
NumMapTask Big-Image.size=BLOCKSIZE+1;
Offset 0; count 1; i 0; k 1;

9: while NumMapTask> 0 do
10:whileOffset<BLOCKSIZE_count do
11: Extract the P-Image[i] from Big-Image. Data according to P-Image[i] Offset;
12: Insert P-Image[i] into GP[k];
13: Offset ← Offset + size of P-Image[i];
14: i ← i + 1;
15: end while
16: k ← k + 1;
17:Allocate GP[k] to the corresponding Map Node;
18: NumMapTask NumMapTask□1; count count+1;
19: end while

Step 1: Produce P-Images. Since that all of the images are typically represented by the structured data after encoding, we firstly produce P-Images. Different from the traditional image processing procedure, P-Image, which is composed of the extracted necessary information, successfully helps to avoid the repeated and time-consuming decoding operation and most importantly, it helps to release memory demanding by storing the numerous P-Images in the hard disk.

Step 2: Produce Big-Image. After completion of Step 1,design a special representation of file called Big-Image to store all of the gained P-Images. Big-Image consists of a data file to store the P-Images and an index file to store the corresponding ID and Offset (Line 4 to 8). Owing to the elegant design of the index structure, Big- Image contributes a lot to rapidly locate the P-Images required for processing. Besides, Big-Image contributes a lot to reduce the disk I/O when compared with conventional small files.

Step 3: Partition Big-Image. The core of our SICP lies in the parallel processing on a cluster of machines by utilizing the computing resources provided by the distributed system. Therefore, the single Big-Image needs to be partitioned into several groups to be processed on the Map Nodes in parallel. The P-Image amount in each group can be set in accordance with the real applications. Search the P-Images stored in the data file via their corresponding Offsets, and then insert these P-Images into one group. (Line 10 to 15).

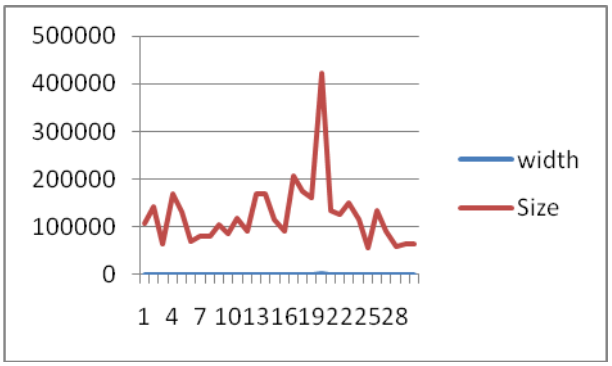

Fig3.4 Comparison between file name and Size

5. Conclusion

SICP is aimed at processing those large-scale image data that have been stored in the distributed system. Decode these static images first to maintain the necessary information as their corresponding P-Images which will be then stored in the data file contained in Big- Image. Then, when image processing is required, we just need to index the index file also stored in Big-Image to find the demanded P-Images which provide the needed image information. Given the needed image information, we can then implement the related image processing algorithms aimed at image classification, retrieval, detection, etc.

References

[1] Le Dong, Member, IEEE, Zhiyu Lin, Yan Liang, Ling He, Ning Zhang, Qi Chen, Xiaochun Cao and Ebroul Izquierdo, Senior Member, "**A Hierarchical Distributed Processing Framework for Big Image Data**" IEEE Transactions on Big Data Volume: 2, Issue: 4, Dec. 1 2016).

[2] F. Wu, Z. Wang, Z. Zhang, Y. Yang, J. Luo, "Weakly Semi-Supervised Deep Learning for Multi-Label Image Annotation," IEEE Transactions on Big Data, vol.1, no.3, pp.109-122, 2015.

[3] L. Dong, J. Su and E. Izquierdo, "Scene-oriented Hierarchical Classification of Blurry and Noisy Images," IEEE Trans. Circuits and Systems for Video Technology, vol.21, no.5, pp.2534-2545, 2012.

[4] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, D.H.J.Epema, "Performance analysis of cloud computing services for many-Tasks scientific computing," IEEE Trans. Parallel and Distributed Systems, vol.22, no.6, pp.931-945, 2011.

[5] Y. Lin, F. Lv, S. H. Zhu, and M. Yanget al, "Large-scale image classification: fast feature extraction and SVM training," CVPR, pp.1689-1696, 2011.

[6] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Communications of the ACM, vol.51, no.1, pp.107-113, 2008.

[7] X. Y. Zhang, L. T. Yang, C. Liu, J. J. Chen , "A Scalable Two-Phase Top- Down Specialization Approach for Data Anonymization Using MapReduce on Cloud," IEEE Trans. Parallel and Distributed Systems, vol.25, no.2, pp.363-373, 2014.

[8] B. Scholkopf, J. Platt, and T. Hofmann, "MapReduce for machine learning on multicore," NIPS, pp.281-288, 2007.

[9] I. Demir and A. Sayar, "Hadoop plugin for distributed and parallel image processing," SIU, pp.1-4, 2012.

[10] I. Palit, and C. K. Reddy, "Scalable and parallel boosting with MapReduce," IEEE Transactions on Knowledge and Data Engineering, vol.24, no.10, pp.249-255, 2009.