Similarity Search in a Very Large Scale Using Hadoop and HBase

Rapport de recherche CEDRIC 2012 –
`http://cedric.cnam.fr`

Stanislav Barton (CNAM), Vlastislav Dohnal (Masaryk University), and Philippe Rigaux (CNAM)

Contents

Abstract

English. We experimented a solution for large-scale indexing of multimedia datasets based on an existing platform, namely Hadoop/HBase. Since this platform does not natively supports indexing of multimedia descriptors in a high-dimensional space, we chose to adopt a solution based on a transformation of the problem to a more traditional Information Retrieval approach, and inspired by research elaborated with our parters. We implemented the solution and conducted extensive experiences to assess its robustness in a Big Data context. The design of this solution and the results obtained so far are described in what follows.

1 Introduction

In order to successfully scale the content based search to very large numbers of data objects, e.g. photographs or images, both proper access structure and infrastructure on which it will run need to be found. In our experiments, the search system forms the disk-based DB access structure that is fully distributed using the MapReduce framework and stored in HBase.

Local description of the data objects makes possible finer grained search within the data object, since more than one description exists for it. For instance, using the local visual description technique, e.g. SIFT or SURF techniques, makes possible to search for particular visual objects over the indexed database – e.g. search for a particular historical monument, that is represented by a set of descriptors. The burden of computational complexity bound with this type of search – one photograph represented by several thousands of local feature vectors – and a multi similarity query consisting of hundreds of individual similarity queries makes this problem very challenging considering again the very large scale.

Therefore, to make large collection of data searchable by content, including the local description techniques, we have conducted the research in two main axes. Firstly, it is tailoring the disk-based DB access structure to be seamlessly distributed via MapReduce – build-up phase, storage and search – is tackled in Section 2. Secondly, addressing the problem of subset type of queries on local descriptor data which is described in Section 3.

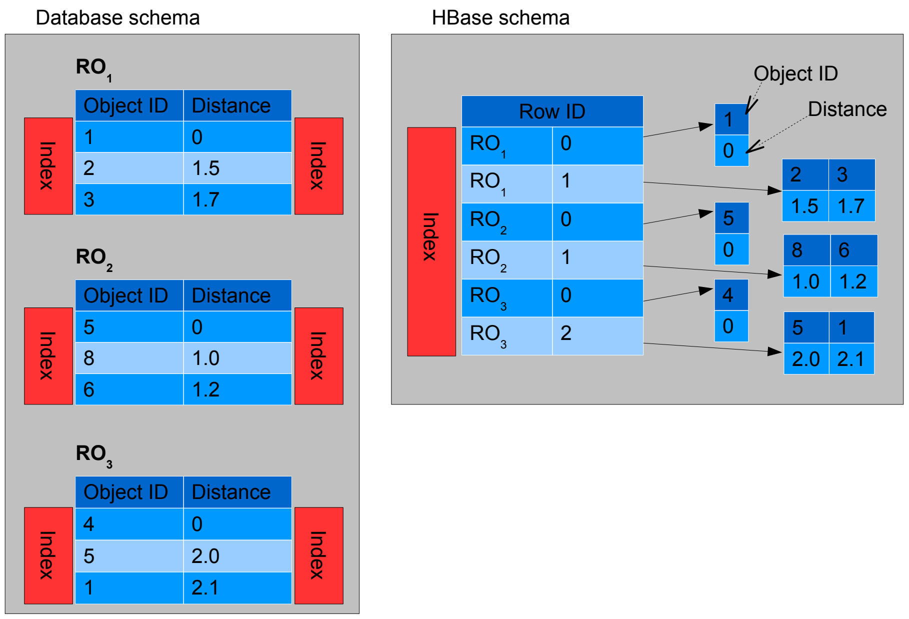

Figure 1: Transformation of the DB schema into an HBase schema.

2 Building Index for Large Scale Amounts of Data Objects

The former design of the disk based access structure proved to have few drawbacks that limited its usage on really large scale data. Firstly, it was the number of insert operations – each precomputed distance was represented as a one insert operation to the database. This imposed a large overhead during the build up phase – one billion of data objects with 8 saved precomputed distances result into 8 billions of DB inserts. Obviously each data base system has its maximum insert throughput per second, thus, the demand needs to be optimized in order to maximize the benefit of these limitations.

Secondly, even though the MapReduce approach to process similarity queries provides easy way to parallelize the query processing, the strict limitation represented by the start of the Reduce phase after the whole Map phase is done proved to be very limiting. Therefore, the alterations to the access structure design, that make possible to process the whole query during the Map phase, are presented.

2.1 Redesigning the Disk-Based DB Access Structure

Since the HBase is not traditional relational database, Figure 1 presents the transformation of the DB schema into an equivalent one usable in HBase. As we mentioned, this schema, from the scalability point of view, presents a flaw in number of insert operation imposed onto the build up phase and the consequent query processing, namely the necessity of computing the inner join. Both these cons are addressed by a redesigned schema and simplified search algorithm.

The former schema has following structure:

Row ID	**Distances**	**Orders**
reference_object_id + distance_slot	data_point_id $\rightarrow$ distance	data_point_id $\rightarrow$ order

The redesigned schema:

Row ID	**Distances to ROs**
reference_object_id + distance	data_point_id+reference_object_id $\rightarrow$ distance

In the redesigned schema, each row contains all precomputed distances for one data object – all distances to closest reference objects for one data object are stored on one row. However they are indexed – using the implicit index on row id – via the shortest distance between data point and its closest reference point. Using such index, it is not necessary to do the inner join for first s closest reference points to the query point, but a certain range is scanned around s closest reference points. These retrieved candidates are then filtered using the distances to the other reference objects stored on one row.

The inner join is substituted by a simple check on the client side – the s closest reference points need to be present in all retrieved rows. This has proven to be more strong pruning paradigm then pruning using only the precomputed distances – especially in data where mutual distances have large mean and small variance. That is attributed to more strict data locality preserving indexing. This property was already present in the former access structure design and forms the most valuable contribution.

This altered schema gives several advantages. It reduces the amount of insert operations – there is only one insert operation per indexed data object (previously there was as many inserts as distances stored for each data object). This proved to be also a major plus for enlarging the index buildup phase throughput. Secondly, the MapReduce query processing can avoid the Shuffle and Reduce phase previously needed to compute the inner join. In this way, the verified candidates retrieved from the HBase can be immediately emitted on the output during the Map phase, if they comply with the pruning/filtering conditions – all the needed information is present on one processed row.

The preliminary results on the 30M SIFT dataset showed, that the index can be fed to the HBase one order of magnitude faster than using the previous schema. As for the searching, the query processing time has diminished about 30%.

2.2 Data Set

In order to test the limits of the proposed innovations, an indexing structure for a dataset comprising of roughly 2 billions of SIFTs is intended to be built. The data set represents roughly million of holiday photos[1] [3] where from each on average 2,000 of SIFTs were extracted. The files represent 235GB of gzipped binary data. From the scalability point of view, If each of the SIFT vector is treated as one data object (global descriptor), the index would make possible to process similarity queries on 2 billions of data objects (photos).

In order to be able to read the data from the MapReduce framework, the data was reformatted and put into a SequenceFile – Hadoop's native binary file format that allows compression on both record or block level. The block-compressed data has 247GB, but proved to be very slow to process – bzip type of compression is very slow – both to compress and decompress. The non-compressed binary data takes up 547GB of space. However, due to the space reasons, the smaller data representation was used to build the index.

2.3 Experience

To build the access structure, some parameters need to be set, especially the number of pivots $|P|$. The formula for the formerly structured index was:

$$|P| = \frac{n \times m}{O}$$

where n is the size of the database, m is fixed as data's intrinsic dimensionality and O is targeted average number of objects in the covering area of one pivot. O actually formed the upper bound on the search algorithm complexity – when inner join is done, this represented average maximum number of retrieved candidates.

Contrarily, in the redesigned schema, the total number of rows in the DB index is no more $n \times m$, but only n. This might imply that with the same

[1]http://lear.inrialpes.fr/~jegou/data.php

complexity bound O, less pivots need to be selected. The resulting numbers for our scalability experiment are $20,000 = \frac{2,000,000,000}{100,000}$.

The diminished number of reference object is convenient, because it is necessary to find m closest from the set for each indexed data object, that even with the use of M-Tree [2], makes the whole process scale rather poorly, even though the performance gain over the sequential scan was one to two orders of magnitude. However, this might be enhanced by using access structure more suitable for this (e.g. AESA [6]) – exact search trading memory for speed.

On the other hand, the number of pivots determined the performance of the access structure in terms of recall and precision, which we expect not go worse than with the previous design. The question is the impact on the overall time to process query, since the amount of data transferred to the client grows, because the filtering done previously by the inner join on the DB server side is now done on the client side.

2.3.1 The Cloud and the Job

To compute the index an infrastructure of MetaCentrum[2], Czech republic was used. In our setting, a cluster of 15 computers was used. All computers are physically placed in one rack and on one switch connected by a Gigabit ethernet. Each computer sports two quadcore Intel Xeon X5570 processors on 2.93GHz, 24GB RAM and two 300GB SAS 15krpm hard drives.

As for the software, we used Hadoop in version 0.20.2, HBase in version 0.20.4, 64-bit Linux and Java 6. The distributed file system (DFS) had replication factor set to three, and block size to 256MB. The Hadoop settings were the same for all nodes in the cluster. One computer ran the NameNode (server) for the DFS, and the JobTracker (server) for the Hadoop platform and was not dedicated. Each node was then able to process seven Map tasks and one Reduce task at one time. This makes 105 Map tasks and 15 Reduce tasks processable by the cloud at one time.

The MapReduce job for the index creation takes on the input the SequenceFile containing the SIFT feature vectors and M-Tree built on the previously randomly selected reference objects. The output is another SequenceFile containing materialized HBase inserts. Not to feed the HBase

[2]The MetaCentrum project, an activity of the CESNET association, covers a majority of activities concerning Grids, super-, cluster- and grid computing and/or high performance computing in general in the Czech Republic. It operates and manages distributed computing infrastructure consisting of computing and storage resources owned by CESNET as well as resources of co-operative academic centres within the Czech Republic.

directly was chosen in order to have a DB dump that would make possible to re-create the index on demand, because it is hard to make a dump of an HBase DB directly.

2.3.2 Performance

The 247GB Sequence file containing the 2 billion SIFT feature vectors takes 745GB of space in the DFS. To copy the input data into the DFS took less than an hour. Recall that the upper complexity bound is $n \times |P| = 2,000,000,000 \times 20,000 = 40,000,000,000,000$ of distance computations. The actual number is lowered by the fact that an access structure was employed to search for the m closest pivots.

On this setting the estimated time to index the whole data set is two days. Recall, that each DFS block represents one Map task in the job. With 256MB block size and 247GB input file, we get about 985 blocks in DFS that make the same amount of Map tasks to process. The measured throughput of the cloud is about 10,500 indexed SIFT points per second, thus, the ETA for the dataset should be 53 hours. This makes the data throughput of about 1.5 MB of compressed data per second. Though, a better values can be achieved by tuning the parameters of the cloud for this job, i.e., enlarging the sort buffers to minimize the disk IO operations throughout the computation.

The whole job took 57 hours and 21 minutes. The Map phase took 51 hours 24 minutes. The average time to do one Reduce task – to copy out the data into the sequence time – took 3 hours and 51 minutes. The result is a sequence file divided into 15 parts each having 68GB – 1TB of data, 3TB in HDFS. These files represent the HBase dump and it takes 12 minutes to copy out each part from the HDFS.

3 Data Object Description via Gaussian Modeling of its Local Descriptors

The conclusions of the previous section foreshadow, that being able to index and consequently search in data collection having billions of objects is possible. Even though the scalability was presented on local visual features, that could be easily substituted by global features. On the other hand, it could be seen that billions of local features usually represent orders of magnitude less actual data objects (e.g. photos, pictures).

Therefore in this section we would like to discuss an approach that would allow treating local features as global ones, i.e., having a compact representation per one data object (photo) and querying by this global compact representation. Lets present a naive approach on the SIFT data: each SIFT feature vector is treated as one object (by the access structure) and then the query for a data object (a photo represented by a set of SIFTs) comprises by a set of similarity queries, one for each SIFT in identified in the query photo. One can easily see, that this might get computationally very intensive on complex queries and large searched collection.

Philbin et al. [5] presented solution to this problem by SIFT quantization and bag-of-words approach. This allows to use the widely known text information retrieval approaches to ease the search complexity. Still, doing this in large scale is a challenging problem – quantization is a clustering problem – and the performance of IR techniques on large dictionaries (demanded by the search accuracy) might deteriorate. Nonetheless, this approach is still the most popular due to its provided usability.

3.1 Multivariate Gaussian Model and Symetrized Kullback-Leibler Distance

As we mentioned, we seek a method to substitute the complete local feature representation of one data object with a more compact representation that will work in a way that global representation does – fixed size and *reasonable* distance computation costs. We have studied several approaches, for instance z-ordering compaction, but the most promising shown to be modeling the SIFT feature vectors of one data object by a multivariate gaussian model. The similarity between such models is consequently measured with symetrized Kullback-Leibler distance (SKLD) [1]. To model several features into one and then using SKLD to measure similarity is used in audio retrieval. However, to apply it to visual local features to represent one data object has not been researched.

The Gaussian multivariate distribution is represented by the vector μ of expected values and the covariance matrix Σ. With SKLD, one can measure the *difference* between the distributions of the models. Basically, on the example of SIFT feature vectors, we should be able to measure the *difference* between the particular photos by measuring the divergence in the distributions of SIFTs that form the two models.

The descriptor created from the SIFT feature vectors of one data object (i.e., photo) is got by computing μ, Σ and Σ^{-1}, where $\mu = [E[X_1], \ldots E[X_{128}]]$, where $E[X_i]$ denotes expected value of the i-th coordinate from the SIFT set. Matrix Σ is a covariance matrix where entry $e_{ij} = Cov[X_i, X_j]$ and Σ^{-1} denotes inverse matrix to Σ that is needed for fast computation of SKLD. Thus the resulting descriptor is represented by a vector having $128 + 2 \times 128 \times 128$ dimensions. In other words, each photo is represented by a vector of a length being equivalent to 257 SIFT feature vectors.

3.1.1 The Ground Truth and Results

In order to test the relevance of the proposed technique following test with two ground truth datasets was conducted. The test comprised of computing the matches of SIFT features between a set of photographs using brute force and L_2 metric. In this test, the threshold on the distance was set, and whenever a distance between a SIFT from one set and a SIFT from other set was under this threshold, it was called a match. The resulting number of matches was normalized by the number of SIFTS in the first set.

We had two sets of photos, queries and their relevant answers. We have computed the match ratios between all queries and all the possible answers. We computed the SKLD between the models representing these objects in a same fashion. After that we have studied the correlation between the corresponding entries in the match ratio and SKLD matrices.

The first data set on which we have conducted this experiment was taken from the evaluation package for the Grenoble data set [3]. It comprises of 500 queries and about 1000 relevant answer objects. We have taken 5 queries with average 3 relevant objects. We have taken all photos as queries and also all photos as possible answers. The resulting average correlation was about 0.7.

This simple proof-of-concept experiment lead us to conduct the evaluation on larger ground truth dataset that would be more statistically representative and also renowned. For this, we have used the data set of photos of various Oxford buildings [5]. The photos were acquired by the authors manually by posing key word queries in Flickr. It comprises of 11 query classes (different buildings), each class comprising of 5 photos (various shots of the same building). Data set also contains annotations of visually relevant answers. The whole data set is 5,500 photos having 1,000,000 SIFT features.

The brute force matches were computed for two distance thresholds – 200 and 250. It takes more then a week to compute this ground truth for all 55 query images and 5,500 image collection. Unfortunately, only partial results are computed at this moment and need to be processed in order to draw precise conclusions.

3.2 Increasing Accuracy

Even though the results on the former data sets are not yet fully interpreted, one phenomenon was observed. This was the on average very high ratio of SIFT point matches between pictures that were definitively visually not similar. These *false true* matches bias the outcomes of both brute force matching and the proposed model similarity approach. In fact, the number of relevant matches between two photos is very small ($<$50 matches), considering that there are thousands of SIFTs in both sets.

We would like to conduct a research in order to detect such distractors – SIFTs that appear in large numbers of pictures (that are not visually similar) and thus are not discriminative, in order to remove them before the model is created. These SIFTs can be compared to the stop words known from text information retrieval. We believe, that in the bag-of-words approach to index local visual features, these stop words are easily detected by their high frequency in the inverted files and not used for actual searching.

3.2.1 Data Object Segmentation

Further impact on the accuracy of the whole approach might have the segmentation of the data object (picture) before computing its model. We propose to verify following approaches:

Z-Order of x and y Coordinates – in one photo, order the sift points according to their z-ordering computed on the x- and y-coordinates. Divide the domain into predefined chunks and compute models on these. One data object would have more than one model representation. The chunks can overlap – sliding window with certain size and step.

Photo Segmentation – segment the photo in a fashion similar to the quad-tree descriptor.

3.3 Increasing Scalability by Cutting the Feature Vector

The resulting vector representing the model on the SIFT data is $128 + 2 \times 128 \times 128 = 65,792$ dimensions. The size of the model description grows quadratically with the size of the original vector. Even though the dimensionality reduction techniques are investigated for the SIFT descriptors, they need to be applied during the extraction phase [4]. Therefore these methods are not applicable directly on already extracted data.

Because the proposed model approach creates a *summary* of a set of feature vectors, we propose to cut the feature vector in several parts and consequently proceed on these parts with the proposed model approach. For example, by cutting each SIFT vector in 8 parts, each covering 16 dimensions, we get 8 models, each represented by 544 dimensional vector – $8 \times 544 = 4,352$ dimensional vector representation.

The question is what the impact is going to be on the accuracy of the model description and also what is the best method to group the original SIFT dimensions, together with the group size.

4 Conclusions and Future Work

The research has been conducted in two major axes. The first axe, very large scale searching in metric spaces and the second one, easing the complexity of searching in local feature descriptions.

In the first axe, we have successfully achieved to create the indexing structure for 2 billions of high dimensional vectors. Though, the performance of the indexing structure will be the subject of the further study. However, the proposed MapReduce approach proven to be very flexible and almost without limits from the scalability point of view. The major bottleneck right now seems to be *k*NN query for *m* closest reference points, whose complexity grows super-linearly with the number of reference objects needed.

In the second axe, the proposed approach seem to be promising and might form an alternative to the current state-of-the-art solution represented by the vector quantization and the bag-of-word approach. In this aspect, the practical comparison of the performance of the proposed method with the state-of-the-art need to be finished.

References

[1] C. Charbuillet, G. Peeters, S. Barton, and V. Gouet-Brunet. A fast algorithm for music search by similarity in large databases based on modified symetrized kullback-leibler divergence. In *Proc. of CBMI'10*, pages 1–6, 2010.

[2] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435, 1997.

[3] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In Andrew Zisserman David Forsyth, Philip Torr, editor, *European Conference on Computer Vision*, volume I of *LNCS*, pages 304–317. Springer, oct 2008.

[4] Yan Ke and R. Sukthankar. Pca-sift: a more distinctive representation for local image descriptors. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 506–513, 2004.

[5] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *IEEE CVPR '07*, pages 1–8, 2007.

[6] E V Ruiz. An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recogn. Lett.*, 4(3):145–157, 1986.