2015 IEEE International Conference on Big Data (Big Data)

Indexing Media Storms on Flink

Dimitrios Rafailidis*, Stefanos Antaris†
*Department of Informatics, Aristotle University of Thessaloniki
†Department of Computer Science, University of Cyprus
draf@csd.auth.gr, antaris.stefanos@cs.ucy.ac.cy

***Abstract*—We propose a media storm indexing algorithm using Map-Reduce in our recently proposed CDVC framework. In this study, CDVC is built on Flink, an open-source platform for stream data processing. The question we answer is how to store massive image collections; for instance, with over one million images per second, as well as with varying incoming rate. In our experiments with two benchmark datasets of 80M and 1B image descriptors, we evaluate the proposed algorithm on different indexing workloads, that is, images that come with high volume and different velocity at the scale of 10^5-10^6 images per second. Using a limited set of computational nodes, we show that we achieve a significant speed up factor of nine, on average, compared to conventional indexing techniques, in all settings. Finally, we make our source code publicly available.**

***Index Terms*—Stream processing, multimedia big data, indexing, cloud computing**

I. Introduction

In this study, we define media storms as large batches of fast incoming images, typically $10^5 - 10^6$ images per second, on average. Nowadays, several real-world applications require an efficient media storm indexing algorithm, for example, the continuously growing and massive image collections that come from social networks [1], or the massive amount of solar image data transmitted by the NASA's satellites [2]. Processing media storms is computational intensive; therefore, the challenge is to support a fast indexing mechanism to store and accurately search over these massive and fast incoming image collections.

In our recent work [3], we introduced CDVC, a framework suitable to handle massive image collections in the cloud. CDVC performs parallel image similarity search in the cloud, by exploiting the Dimensions' Value Cardinalities (DVC) of the image descriptors. DVC are defined as the number of discrete values occurred in a specific dimension throughout a set of image descriptor vectors [3]. Compared to state-of-the-art distributed similarity search strategies, such as parallel hashing [4] and multiple KD-Trees [5], the big advantages of CDVC are low preprocessing cost and high search accuracy in low time, because CDVC avoids to construct complex index structures and efficiently spits the computational effort into multiple computational nodes. Although the CDVC's indexing mechanism is efficient by being independent of the scale of the already stored images (Section II-A), it cannot handle the massive and fast incoming media storms, because new images are indexed sequentially [3]. The challenges of indexing media storms are: (i) to correctly identify their logical positions in the CDVC index, that is, a double linked list complying with the DVC-based strategy, preserving the search accuracy high and the search time low in the query processing; and (ii) to keep the computational cost of the media storm indexing mechanism low. Meanwhile, there are several methods that efficiently index media storms, such as the work of [1]; however, the complex index structures of KD-trees are used, having thus high computational cost in the online search [3]. Therefore, in this study we focus on the CDVC framework.

In this paper, we propose a parallel media storm indexing algorithm using Map-Reduce in the CDVC framework, which is built on Flink[1], an open-source platform for stream data processing. The reasons for selecting Flink, instead of other platforms, such as Spark[2], are because Flink provides a better resource usage of the cluster when recovery is required [6], as well as, Flink provides incremental iterations in its dataflow model [7]. In our experiments we evaluate the performance of the proposed media storm indexing algorithm on different indexing workloads within several "storming time frames", that is, the duration that the storm lasts.

II. Proposed Media Storm Indexing Algorithm

A. Preliminaries and Problem Definition

Preprocessing: According to [3], given M computational nodes and N descriptor vectors with D dimensions, CDVC performs a multi-sort algorithm in parallel. The dimensions of image descriptors with high DVC are prioritized, assuming that these dimensions are more discriminative. The priorities based on DVC are stored in a priority index vector $\mathbf{p}$. After the multi-sort algorithm has finished, the logical positions of the descriptors are stored in a double linked list $\mathbf{L}$, which is the index of CDVC. Following [3], in this study, we assume that the descriptors have been extracted locally and not in the cloud. Nevertheless, several works follow parallel strategies to speed up the extraction of descriptors.

Indexing: Given a new descriptor vector $\mathbf{v}_q$, the goal is to identify the position pos_q in the double linked list $\mathbf{L}$ based on the priority index $\mathbf{p}$. The correct position pos_q is located by identifying the set $\mathcal{V}_{pk}$ ($|\mathcal{V}_{pk}| \ll N$), which is the set of the already stored image descriptors that have the same value with the new descriptor $\mathbf{v}_q$ at the first sorted dimension. We set a primary key pk to the value of the first sorted dimension, corresponding to the dimension with the highest DVC. Then, the new descriptor is inserted in the logical position pos_q in $O(|\mathcal{V}_{pk}| \cdot \log|\mathcal{V}_{pk}|) + O(D)$, updating the double linked list

[1]https://flink.apache.org
[2]http://spark.apache.org

from $\mathbf{L}$ to $\mathbf{L}'$. Since $|\mathcal{V}_{pk}| \ll N$, the indexing algorithm is independent of the dataset size N [3].

Query Processing: Given the updated double linked list $\mathbf{L}'$ and the position pos_q of the image query q, a set $\mathcal{V}_{2W}$ of image descriptors is generated which is located in the updated double linked list $\mathbf{L}'$ in W previous and W next to the position pos_q, corresponding to a search radius $2W$. To retrieve the top-k results, the distances between the set $\mathcal{V}_{2W}$ and the query image descriptor vector $\mathbf{v}_q$ are calculated using M computational nodes with T parallel threads.

Problem Definition: We consider media storms as a consecutive sequence of batches, where each batch contains a set $\mathcal{X}$ of incoming image descriptors at a certain time step within the storming time frame. The problem of indexing media storms in the CDVC framework is *"to correctly index each set $\mathcal{X}$ of the incoming descriptors in parallel, and to update the double linked list from $\mathbf{L}$ to $\mathbf{L}'$, by preserving the multi-sorted logical positions based on the priority index $\mathbf{p}$".*

B. Algorithm

The proposed approach is presented in Algorithm 1.

Algorithm 1: Media Storm Indexing Algorithm

```
   Input: (1) $\mathcal{X}$, (2) $\mathbf{p}$
   Output: $\mathbf{L}'$
1  MAP 1
2  Input : (1) pairs $< \mathbf{v}_i, \mathbf{p} >$, with $\mathbf{v}_i \in \mathcal{X}$, (2) $\mathbf{p}$
3  Method
4      $\mathbf{v}'_i \leftarrow$ Sort $\mathbf{v}_i$ based on $\mathbf{p}$
5      $pk \leftarrow v'_{i1}$
6      Emit $< pk, \mathbf{v}'_i >$
7  COMBINE
8  Input : pairs $< pk, \mathbf{v}'_1 >, < pk, \mathbf{v}'_2 >, \ldots, < pk, \mathbf{v}'_{|\mathcal{X}|} >$
9  Method
10     $\mathcal{X}_{pk} \leftarrow$ group $\mathbf{v}'_1, \mathbf{v}'_2, \ldots, \mathbf{v}'_{|\mathcal{X}|}$ with the same
11             primary key $pk$
12     Emit $< pk, \mathcal{X}_{pk} >$
13 MAP 2
14 Input: (1) pairs $< pk, < \mathbf{v}'_1, \mathbf{v}'_2, \ldots, \mathbf{v}'_{|\mathcal{X}_{pk}|} >>$, (2) $\mathcal{V}_{pk}$
15 Method
16     $\mathbf{L}^{(m)} \leftarrow$ sort $< \mathbf{v}'_1, \mathbf{v}'_2, \ldots, \mathbf{v}'_{|\mathcal{X}_{pk}|} >$ with $\mathcal{V}_{pk}$
17     Emit <key, $\mathbf{L}^{(m)}$ >
18 REDUCE
19 Input :<key, $< \mathbf{L}^{(1)}, \mathbf{L}^{(2)}, \mathbf{L}^{(3)}, \ldots, \mathbf{L}^{(m)} >>$
20 Method
21     $\mathbf{L}' \leftarrow$ Merge $< \mathbf{L}^{(1)}, \mathbf{L}^{(2)}, \mathbf{L}^{(3)}, \ldots, \mathbf{L}^{(m)} >$
22     Emit <key, $\mathbf{L}'$ >
```

MAP 1: The batch $\mathcal{X}$ of the incoming descriptors is divided into M computational nodes. Each node takes as input a descriptor $\mathbf{v}_i \in \mathcal{X}$ and the priority index $\mathbf{p}$ (line 2). Then, to comply with the DVC-based strategy, the descriptor $\mathbf{v}_i$ is reordered based on $\mathbf{p}$ (line 4). The value of the first dimension of the descriptor v'_{i1} is set as a primary key pk (line 5). The output of the MAP 1 phase is a pair $< pk, \mathbf{v}'_i >$, considering the primary key pk as key, and the reordered descriptor $\mathbf{v}'_i$ as value. The total complexity of the MAP 1 phase is $O\left(\frac{|\mathcal{X}| \cdot D}{M}\right)$.

COMBINE: In this phase, the goal is to group the reordered descriptors with the same primary key pk in the same set $\mathcal{X}_{pk} \subseteq \mathcal{X}$ (lines 10-11), resulting in complexity $O(|\mathcal{X}|)$. The output of the COMBINE phase is a pair $< pk, \mathcal{X}_{pk} >$, with pk as key, and the set $\mathcal{X}_{pk}$ as value, that is, the subset of the incoming descriptors that have the same primary key pk.

MAP 2: For each primary key pk, the m-th computational node, with $m \in 1 \ldots M$, fetches the sets $\mathcal{X}_{pk}$ from the COMBINE phase, as well as the set $\mathcal{V}_{pk}$ from the CDVC's distributed databases, that is, the set of the already stored descriptors that also have the primary key pk. Then, the m-th node compares and reorders the subset $\mathcal{X}_{pk}$ of the incoming descriptors along with the subset $\mathcal{V}_{pk}$ of the already stored descriptors, in order to generate the double linked sublist $\mathbf{L}^{(m)}$ (line 17) in $O\left(\frac{|\mathcal{X}| \cdot |\mathcal{V}_{pk}|}{|\mathcal{X}_{pk}| \cdot M}\right)$. Finally, we set the same key to all the M different computational nodes' outputs, to process all the output pairs by a single reducer.

REDUCE: The M different double linked sublists $\mathbf{L}^{(m)}$, produced by the MAP 2 phase, are merged to update the global double linked list from $\mathbf{L}$ to $\mathbf{L}'$, thus indexing the incoming descriptors of set $\mathcal{X}$, in $O(M)$. Summarizing, the total complexity of the proposed media storm algorithm is

$$O\left(\frac{|\mathcal{X}| \cdot D}{M}\right) + O(|\mathcal{X}|) + O\left(\frac{|\mathcal{X}| \cdot |\mathcal{V}_{pk}|}{|\mathcal{X}_{pk}| \cdot M}\right) + O(M) \quad (1)$$

III. Experiments

In our experiments we used GIST-80M (384d)[3] and SIFT-1B (128d)[4], two among the largest benchmark datasets. To evaluate the performance of our media storm indexing algorithm, we first preprocessed the 80% of each dataset, and we considered the remaining 20% as the media storm size, corresponding to 16M and 200M images for GIST-80M and SIFT-1B, respectively. We measure media storms in terms of Incoming Image Rate (IIR) and we evaluate the proposed algorithm in terms of Indexing Speedup Factor (ISF), which are formally defined as follows: *"**IIR** is the number of incoming images per second"*, and *"**ISF** is the ratio of the required time of CDVC's sequential indexing method to the respective time of the proposed media storm indexing algorithm"*. We varied IIR according to the: (i) Logarithmic, (ii) Random, (iii) Exponential and (iv) Normal distributions. The reason for selecting these distributions is their different characteristics, such as different bursting frequency and magnitude. We varied the storming time frame, that is, the media storm duration, in 60, 120, 180, 240 and 300 seconds. Due to lack of space, in Figure 1, we present two representative examples, that is, the Logarithmic and Normal distributions of IIR in GIST-80M.

We ran our experiments on a Flink cluster with 40 nodes, by setting the default main Flink parameters, that is, the JVM heap size, the number of Task Slots, and the number of Buffers, equal to 512, 1, and 2048, respectively. We used the Apache Kafka publish-subscribe messaging system, and the image data were stored and retrieved using HBase. In total we utilized 114 Cores, 228 GB Ram and 2.5TB Hard Disk Storage. Our source code is publicly available[5].

[3]http://horatio.cs.nyu.edu/mit/tiny/data/index.html
[4]http://corpus-texmex.irisa.fr/
[5]http://delab.csd.auth.gr/˜draf/MediaStorms.zip

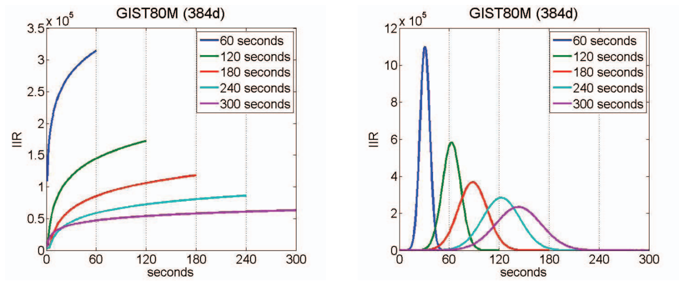

Fig. 1. Logarithmic and Normal distributions of IIR in GIST-80M ($\times 10^5$ incoming images per second). In SIFT-1B, similar IIR are generated with $\times 10^6$ incoming images per second.

In Table I, we report the average and standard deviation of IIR and ISF for the different workloads, corresponding to different distributions and storming time frames. For the same storming time frame, the average IIR is preserved, because the same amount of incoming image descriptors has to be indexed in each dataset; however the standard deviations vary according to the IIR distributions. ISF is increased for larger storming time frames, as the indexing workload is reduced, denoted by lower IIR values. When a media storm's IIR follows a normal distribution, the ISF is preserved for different time frames. This happens because the normal distribution has a pick/burst of IIR, denoted by the higher standard deviation compared with the standard deviations of the logarithmic, random, and exponential IIR distributions, for the same storming time frame. Summarizing, in all settings we achieve 9.95 and 9.34 ISFs, on average, for GIST-80M and SIFT-1B, respectively.

TABLE I
AVERAGE INDEXING WORKLOADS AND SPEED UP FACTORS IN TERMS OF IIR AND ISF, RESPECTIVELY, FOR DIFFERENT STORMING TIME FRAMES.

	GIST80M (384d)		SIFT1B (128d)	
Storming Time Frame (sec)	**Av. IIR** (10^5) $\pm$ **Std**	**Av. ISF** $\pm$ **Std**	**Av. IIR** (10^6) $\pm$ **Std**	**Av. ISF** $\pm$ **Std**
Logarithmic				
60	2.66 ± 0.45	4.97 ± 0.52	3.33 ± 0.45	5.13 ± 0.05
120	1.33 ± 0.35	9.23 ± 0.29	1.66 ± 0.37	7.58 ± 0.08
180	0.88 ± 0.26	14.75 ± 0.02	1.11 ± 0.27	9.14 ± 0.12
240	0.66 ± 0.18	16.12 ± 0.27	0.83 ± 0.18	12.53 ± 0.12
300	0.53 ± 0.09	20.14 ± 0.34	0.66 ± 0.09	15.06 ± 0.19
Random				
60	2.66 ± 0.93	4.15 ± 0.12	3.33 ± 2.21	4.59 ± 0.13
120	1.33 ± 0.36	6.15 ± 0.25	1.66 ± 1.75	6.01 ± 0.09
180	0.88 ± 0.22	8.26 ± 0.42	1.11 ± 0.31	8.94 ± 0.02
240	0.66 ± 0.20	10.16 ± 0.17	0.83 ± 0.14	10.11 ± 0.07
300	0.53 ± 0.20	12.23 ± 0.29	0.66 ± 0.37	11.07 ± 0.48
Exponential				
60	2.66 ± 2.73	3.96 ± 0.12	3.33 ± 2.55	4.85 ± 0.25
120	1.33 ± 1.32	5.17 ± 0.15	1.66 ± 1.73	6.72 ± 0.16
180	0.88 ± 0.78	7.05 ± 0.06	1.11 ± 0.73	8.23 ± 0.12
240	0.66 ± 0.66	9.15 ± 0.02	0.83 ± 0.53	9.76 ± 0.02
300	0.53 ± 0.44	10.11 ± 0.16	0.66 ± 0.37	11.07 ± 0.48
Normal				
60	2.66 ± 3.72	10.25 ± 0.57	3.33 ± 5.18	9.11 ± 0.05
120	1.33 ± 1.93	11.13 ± 0.25	1.66 ± 2.54	10.43 ± 0.12
180	0.88 ± 1.23	11.52 ± 0.12	1.11 ± 1.67	11.82 ± 0.21
240	0.66 ± 0.94	12.07 ± 0.58	0.83 ± 1.25	11.99 ± 0.24
300	0.53 ± 0.77	12.35 ± 0.46	0.66 ± 1.02	12.35 ± 0.29

To verify that the media storms have been correctly indexed in CDVC, we evaluate the search accuracy of CDVC by indexing media storms with the sequential indexing algorithm of [3], and the proposed approach. For the same search radius $2W$, we verified that CDVC's high accuracy is preserved, because the proposed indexing algorithm also correctly identifies the positions of the media storm's images in the double linked list $\mathbf{L}$, as the sequential approach does. In addition, in Figure 2, we present the average total execution time and latency for the query processing, after the media storms have been indexed, by varying the search radius $2W$. Over the variation of $2W$, the latency corresponds to 80-90% of the total execution time.

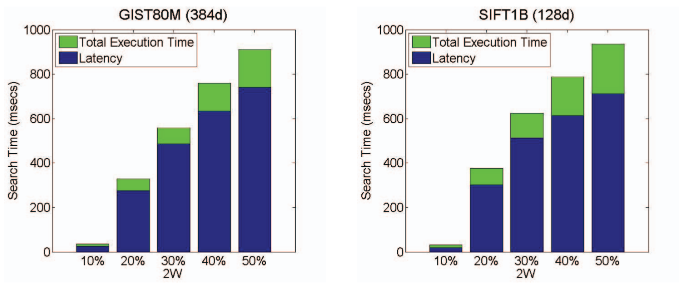

Fig. 2. Computational search time by varying the search radius $2W$.

IV. Conclusion

We presented an efficient media storm indexing algorithm in the CDVC framework, built on Flink. We showed that we achieve a high indexing speed up factor (ISF) in all settings. Interesting future directions are to evaluate the influence of the different parameters of the Flink platform, to perform an experimental evaluation on different stream data processing platforms, such as Spark, and to evaluate the proposed approach in a real-case study, where the media storm distribution will match better with the examined application, for example, we plan to consider the indexing workload that the incoming images in social media platforms create within a certain storming time frame, like the rate with which users upload photos from popular events.

Acknowledgments: This research was conducted during Stefanos Antaris Master studies at the Aristotle University of Thessaloniki.

References

[1] D. Moise, D. Shestakov, G. T. Gudmundsson, and L. Amsaleg, "Terabyte-scale image similarity search: Experience and best practice," in *Proceedings IEEE International Conference on Big Data*, 2013, pp. 674–682.

[2] J. M. Banda and R. A. Angryk, "Scalable solar image retrieval with lucene," in *Proceedings IEEE International Conference on Big Data*, 2014, pp. 11–17.

[3] S. Antaris and D. Rafailidis, "Similarity search over the cloud based on image descriptors' dimensions value cardinalities," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 11, no. 4, pp. 51:1–51:23, Jun. 2015.

[4] M. Norouzi, A. Punjani, and D. J. Fleet, "Fast exact search in hamming space with multi-index hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 6, pp. 1107–1119, 2014.

[5] M. Muja and D. G. Lowe, "Scalable nearest neighbour algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.

[6] S. Schelter, S. Ewen, K. Tzoumas, and V. Markl, ""all roads lead to rome": optimistic recovery for distributed iterative data processing," in *22nd ACM International Conference on Information and Knowledge Management, CIKM'13*, 2013, pp. 1919–1928.

[7] A. Alexandrov, R. Bergmann, S. Ewen, J.-C. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. J. Sax, S. Schelter, M. Höger, K. Tzoumas, and D. Warneke, "The stratosphere platform for big data analytics," *The VLDB Journal*, vol. 23, no. 6, pp. 939–964, Dec. 2014.