Big Data and Web: An Efficient Algorithm Design for DISC

Mahesh S Nayak[1], Dr. M. Hanumanthappa[2], Dr. B R Prakash[3], Dattasmita HV[4]

[1]*Research and Development Centre Bharathiar University, Coimbatore – 641 046, India*
[2]*Professor, Department of Computer Science & Applications, Bangalore University, Bangalore, India*
[3]*Assistant Professor, Department of MCA, Sri Siddhartha Institute of Technology, Tumkur, India*
[4]*Assistant Professor, Govt. First Grade College, Tumkur, India*

***Abstract*: - Data-intensive computing is a paradigm to address the data gap and a platform to allow the advancement in research to process massive amounts of data and implement such applications which previously analyzed to be impractical or infeasible.The existing one-pass analytics algorithm observed to be data-intensive and contrarily requires the ability to efficiently process high volumes of data. MapReduce is supposed to be a programming model for processing large datasets using a cluster of machines. However, the existing MapReduce model is considerably not well-suited for high volume trimmer data, since it is towards batch processing and requires the data set to be fully loaded into the cluster before running analytical queries. This paper examines, from anefficiency standpoint, what the architectural design changes are necessary to bring the benefits of the MapReduce model and streaming algorithm to incremental, the existing MR algorithms.**

I. INTRODUCTION

"Data-intensive Scalable computing is a class of parallel computing applications that use- data parallel approach to process terabytes or petabytes of data and hence represented as big data. The computing applications are deemed according to compute-intensive and data-intensive based on the type of computational requirements and data volumes".[1]

"The advent of the Internet and World Wide Web has given the reason for storing of large amount of information and presenting them online.The business and government organizations create large amounts of both structured and unstructured information which needs to be processed, analyzed, and linked. An IDC white paper sponsored by EMC Corporation estimated the amount of information currently stored in a digital form in 2007 at 281 Exabyte's and the overall compound growth rate at 57% with information in organizations growing at even a faster rate".[3] "The storing, managing, accessing, and processing of this vast amount of data represents a fundamental need and an immense challenge in order to satisfy needs to Search-Analyze-Mine-Visualize[SAMV] this data as information".[5]

"The real-time analytics on large andconcurrent datasets has become an essential challenge to meet the enterprise needs. Like traditional warehouse applications, real-time analytics, using incremental one-pass processing tends to be data-intensive in nature and requires the ability to collect and analyze enormous datasets efficiently. At the same time, MapReduce has emerged as a popular model for parallel processing of large datasets using a commodity cluster of machines. The key benefits of the model are that, it harnesses compute and I/O parallelism on commodity hardware and can easily scale as the datasets grow in size. However, the MapReduce model is not well-suited for incremental one-pass analytics since it is primarily designed for batch processing of queries on large datasets".[6]

"Recently, three Google researchers summarized the data-driven philosophy in an essay titled The Unreasonable Effectiveness of Data.[7] Why is this so? It boils down to the fact that language in the wild, just like human behavior in general, is messy. Unlike, say, the interaction of subatomic particles, human *use* of language is not constrained by succinct, universal "laws of grammar". There are of course rules that govern the formation of words and sentences—for example, that verbs appear before objects in English, and that subjects and verbs must agree in number in many languages—but real-world language is affected by a multitude of other factors as well: people invent new words and phrases all the time, authors occasionally make mistakes, groups of individuals write within a shared context, etc. The Argentine writer Jorge Luis Borges wrote a famous allegorical one-paragraph story about a fictional society in which the art of cartography had gotten so advanced that their maps were as big as the lands they were describing. The world, he would say, is the best description of itself. In the same way, the more observations we gather about language use, the more accurate a description we have of language itself. This, in turn, translates into more effective algorithms and systems".[8]

"Data represent the rising tide that lifts all boats—more data lead to better algorithms and systems for solving real-world problems. Let's start with the obvious observation: data intensive processing is beyond the capability of any individual machine and requires clusters—which mean that large-data problems are fundamentally about organizing computations on dozens, hundreds, or even thousands of machines. This is

exactly what MapReduce does, and the rest of this paperis a platform to present the same". [8]

II. ANALYSIS- BEHIND MAPREDUCE

There are discussions and analyses carried out for the Search-Analyze-Mine-Visualize [SAMV] of large-data problems. The abstract algorithm requires a distinct approach which can optimize the traditional models of computing.

2.1 Scaling

For data-intensivecontext, contrary to the small number of high-end servers, low-end-servers are considered. The symmetric multiprocessing machines stand up to be costlier with large amount of shared memory, which justifies being no cost efficient. "A survey of five thousand Google servers over a six-month period shows that serversoperate most of the time at between 10% and 50% utilization[8], which is an energyinefficient operating region".Furthermore, Datacenter efficiency is a challenge to address on the scaling of data-intensive computing.[8]

2.2 Movement

The high-performance computing algorithms have "processing nodes" and "storage nodes" linked together over high-capacity interconnections. Most of the data-intensive algorithms are less efficient in processorutilization, literally means to have no separation between the compute and storage in the network. In the contrary, MapReduce assumes to represent an architecture which has the processors and storage co-located. Such scenarios justify the responsibility of the distributed file system for managing the data- over the MapReduce.

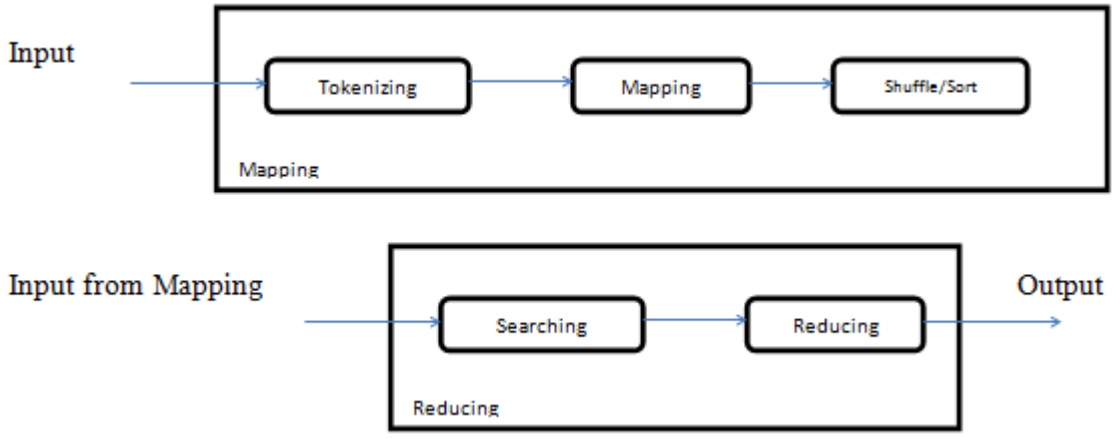
2.3 Sequential negating random access.

A challenge to store the relevant datasets onto memory gives rise to a question, what is the efficiency of the Seektime? TheData-intensive processing literally means to imbibe the fundamentality of memory store to be sequential and having no technique of random access which could impact the efficiency. Furthermore, during the random access, the efficiency of theread headsis mostly negotiable to be zero. Hence, it is advisableto avoid random data access. "A simple scenario [9] poignantly illustrates the large performance gap between sequential operations and random seeks: assume a 1 terabyte database containing 10^{10} 100-byte records. Given reasonable assumptions about disk latency and throughput, a back-of-the-envelope calculation will show that updating 1% of the records (by accessing and then mutating each record) will take about a month on a single machine. On the other hand, if one simply reads the entire database and rewrites all the records (mutating those that need updating), the process would finish in under a work day on a single machine. Sequential data access is, literally, orders of magnitude faster than random data access".[8] [10]

III. BACKGROUND

3.1 The Map Reduce Algorithm

The MapReduce algorithm consists of the tasks Mapping and Reducing.Input is passed onto the Mapper instance post which the syntheses of the input into tokens are done. The tokens are then mapped following the shuffling and sorting of the matching pairs is done. Now the Reduction task includes the searching and reducing the matching pairs or data.



A MapReduce algorithm assists in sending the Map & Reduce the tasks to manageable servers in a cluster.The mathematical algorithms include the following,

- **Sorting:** One of the basic MapReduce algorithms to process and analyze data. MapReduce implements sorting algorithm and thus automatically sort the output key-value pairs from the mapper based on the keys.
- **Searching:** Searching helps in the phases-Combiner (optional) and Reduce
- **Indexing:** The indexing technique which is basically used in the MapReduce is termed as inverted index. Search engines like Google and Bing uses

inverted indexing technique as it has batch indexing technique, the procedural technique profound to be optimal. It is based on A Mapper implementation.

- **TF-IDF[**Term Frequency − Inverse Document Frequency**]:**It is a text processing algorithm and a common web analysis algorithm. The term frequency refers to the occurrence of term in a document.

3.2 Streaming

It is a fundamental model for computations on massive datasets (Alon et al., 1999; Henzinger et al., 1998). The earlier design of the model depicts a defined number of passes on the data and a poly-logarithmic space of size n. In the semi-streaming model, a logarithmic number of passes and*O(n_polylog n)* space (Feigenbaum et al., 2005) were allowed. A comparison to these models and MapReduce would depict a mere difference with regard to the Model of Computation. Feldman et al. (2007) explore the relationship between streaming and MapReduce algorithms. Streaming is used in a wide range of data-intensive oriented applications, where the nature of data is transient data stream rather than persistent. Over the wide range of scope, few applications include, financial applications, network monitoring, security and sensor networks. To define a data stream, literally, it is a continuous, ordered sequence of items. Data streams differ from the traditional batch model in several ways:

- The online arrival of data in the Stream;
- Is the order of the Item a prime component over efficiency?
- A mere dependency of streams and size;
- Post processing activity: Discarded or archived;

The last fact implies that, items cannot be retrieved easily unless they are explicitly stored in memory, which is usually small compared to the size of the input data streams.

IV. PROBLEM ANALYSIS

The Similarity self-join computes pairs of objects in a collection holding a criterion of having the valuesimilaritythat satisfies a defined condition. For instance, in identifying the users based on the category or patronized images in a particular set viz., animals. Identifiable problem in similarity self-join is, on assumption- Consider the collection of objects having the criteria viz., Medical. The aim is to identify the mostly similar objects according to the similar function, say- dental. The task of discoveringsimilar objects within a given collection is common to many real worlddata mining and machine learning problems.

The recommendations by Item-based and user-based algorithms mostly require computing on pair-wise and similarity among users or items. Since the count of users and objects would be large, the similarity scores are usually computed off-line.Near duplicate detection is commonly performed as a pre-processing step before building a document index. On identification, it can be even used to detect the redundant document,which can therefore be removed or it can even be used as content farms and spam websitesexploiting content repurposing strategies.Near duplicate detection finds application also in the area of copyright protection as a tool for discovering plagiarism

Density-based clustering algorithms like DBSCAN (Ester et al., 1996) or OPTICS (Ankerst et al., 1999) inherently join the input data based on similarity relationships. Correlation clustering (Bansal et al., 2004) uses similarity relationship between the objects instead of the actual representation of the objects. All these algorithms will draw high benefit from efficient and scalable MapReduce implementations of similarity joins.

4.1 Analysis of Existing MR Algorithm

4.1.1 MapReduce Term-Filtering (ELSA)

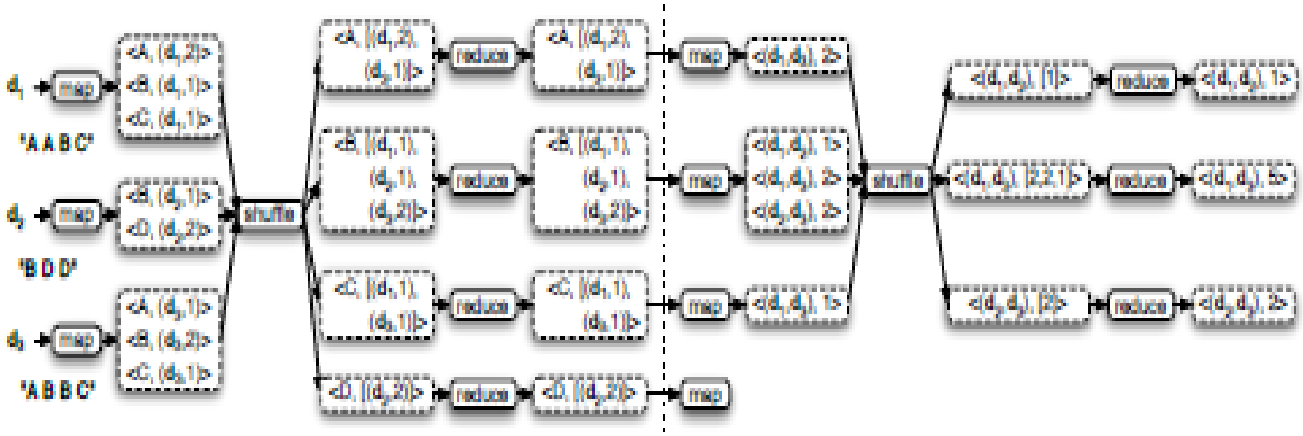
"Elsayed et al. (2008) present a MapReduce implementation of the Term-Filtering method. "The algorithm runs two consecutive MR jobs, the firstbuilds an inverted index and the second computes the similarities".[25]

"**Indexing**: Given a document d_i, for each term, the mapper emits the termas the key, and a tuple $<i; d_i[t]>$ consisting of document ID and weight asthe value. The shuffle phase of MR groups these tuples by term anddelivers these inverted lists to the reducers that write them to disk."[25]

Map: $<i, d_i>$ ⟶ $[<t; <i, d_i[t]>>| d_i[t] > 0]$
Reduce: $<t, [<i, d_i[t]>,<j, d_j [t]>,\ldots]>$ ⟶ $[<t, [<i, d_i[t]>,<j, d_j [t]>,]>]$

"Similarity: Given the inverted list of term t, the mapper produces the contribution w_{ij} [t] = d_i[t]. d_j [t] for every pair of documents where theterm t co-occurs. This value is associated with a key consisting of the pairof document IDs $<h_i,j_{i>}$. For any document pair the shuffle phase will passto the reducer the contribution list $W_{ij} = \{w_{ij} [t] |w_{ij} [t] > 0; \forall t \in L\}$from the various terms, which simply need to be summed up."[25]

Map: <t,[<i,d_i[t]>,<j,d_j [t]>,….]> ⟶ [<<i.j>,w_{ij}[t]>]
Reduce: <<i, j>,w_{ij}> ⟶ [<<i, j>,σ(d_i,d_j) = $\sum_{w \in W ij}^{n} w$>

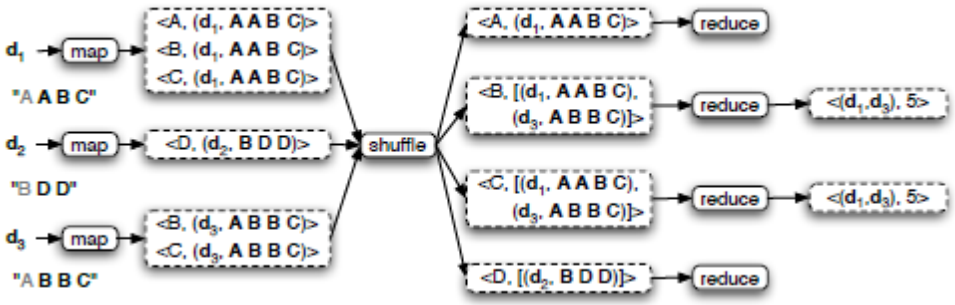


"For ease of explanation, the document vectors are not normalized. Term-Filteringavoids computing similarity scores of documents that do not share anyterm. For the special case in which an inverted list contains only one document,the similarity Map function does not produce any output. Thetwo main problems of ELSA result evident from looking at the image.First, long inverted lists may produce a load imbalance and slow downthe algorithm considerably, as for term "B" in the figure. Second, thealgorithm computes low similarity scores which are not useful for thetypical applications, as for document pair <d1; d2> in the figure."[25]

4.1.2 MapReduce Prefix-Filtering (VERN)

"Vernica et al. (2010) present a MapReduce algorithm based on Prefix- Filtering that uses only one MR. For each term in the signature of a document t €S(d_i) as defined by Prefix-Filtering, the map function outputs a tuple with key the term t itself and value the whole document d_i. The shuffle phase delivers to each reducer a small sub-collection of documents that share at least one term in their signatures. This process can be thought as the creation of an inverted index of the signatures, where each posting is the document itself rather than a simple ID. Finally, each reducer finds similar pairs among candidates by using state-of-the-art serial algorithms (Xiao et al., 2008). The Map and Reduce functions are as follows"."[25]

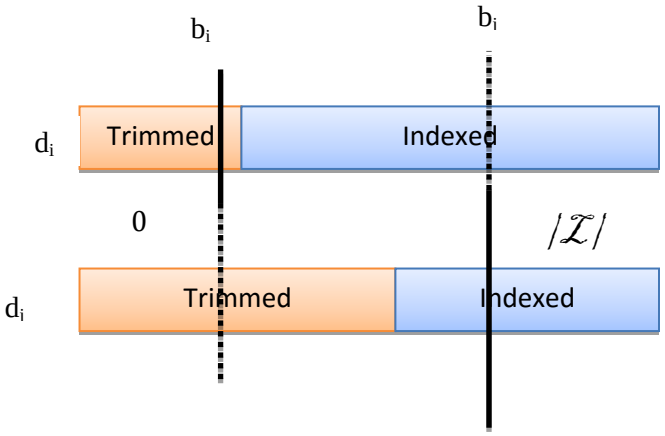
Map: <i, d_i> ⟶ [<t, d_i>|t€S(d_i)]
Reduce: <t,D_v=[d_i,d_j,…]> ⟶ [<<i, j>, σ(d_i,d_j) >>|d_i,d_j€D_v]



"Light gray terms aretrimmed from the document by using Prefix-Filtering. Each document isreplicated once for each non-trimmed term it contains. Finally, the reducercomputes the similarity of the bag of documents it receives by employinga serial SSJ algorithm. VERN computes thesimilarity of the pair <d1; d3> multiple times at different reducers".[25]

4.2 Proposed Algorithm

The proposed system would be an extension of the ELSA algorithm. The algorithm includes the indexing phase following the computational phase. The proposed algorithm would shorten the inverted lists by employing Prefix-Filtering .The effect of Prefix-Filtering is to reduce the portion of document indexed. The terms occurring in d_i up to position b (d_i), or b_i for short, need not be indexed. By sorting terms in decreasing order of frequency, the most frequent terms are discarded. This trimming shortens the longest inverted lists and brings a significant performance gain.



4.2.1 Cost analysis:

Indeed, estimating the cost of a MapReduce algorithm is quite difficult, because of the inherent parallelism, the hidden cost of the shuffling phase, the overlap among computation and communication managed implicitly by the framework, the non-determinism introduced by combiners and so on. The proposed model separately, the three main steps of a MapReduce jobs: the Map and Reduce functions, and the volume of the data to be shuffled. In particular, the cost associated to the function instance with the largest input is considered.

V. CONCLUSION

The paper brings an analysis of the architectural and theoretical understanding of the problem and the proposed algorithm which helps in the load analysis and reduction. The empirical and theoretical analyses showed that there are limitations with present algorithm which can be optimized to handle the high volume indexed data. The advanced data analysis platform that employs a purely index-trim-based framework, with various techniques to enable incremental processing and fast in-memory processing for frequent keys is proposed. In future work, the proposed system can be extended to support a wider range of incremental computation tasks with minimized I/O, online aggregation with early approximate answers, and stream query processing with window operations.

REFERENCES

[1]. A.M. Middleton. "Data-Intensive Technologies for Cloud Computing," Handbook of Cloud Computing. Springer, 2010.
[2]. Vinton Cerf. *An Information Avalanche*. IEEE Computer, Vol. 40, No. 1, 2007, pp. 104-105.
[3]. J.F. Gantz, D. Reinsel, C. Chute, W. Schlichting, J. McArthur, S. Minton, J. Xheneti, A. Toncheva, and A. Manfrediz, IDC. *The Expanding Digital Universe Archived* , March 10, 2013, at the Wayback Machine, White Paper, 2007.
[4]. P. Lyman, and H.R. Varian.*How Much Information?* 2003, University of California at Berkeley, Research Report, 2003.
[5]. F. Berman. *Got Data? A Guide to Data Preservation in the Information Age*, by Communications of the ACM, Vol. 51, No. 12, 2008, pp. 50-56.
[6]. Boduo Li, Edward Mazur, YanleiDiao, Andrew McGregor, Prashant Shenoy. *A Platform for Scalable One-Pass Analytics using MapReduce*,URL: https://people.cs.umass.edu/~mcgregor/ papers/11-sigmod.pdf
[7]. Alon Halevy, Peter Norvig, and Fernando Pereira. *The unreasonable effectiveness of data*, Communications of the ACM, 24(2):8–12, 2009.
[8]. Jimmy Lin and Chris Dyer. *Data-Intensive Text Processing with MapReduce*, April 11, 2010. URL: https://lintool.github.io/MapReduceAlgorithms/MapReduce-book-final.pdf.
[9]. Arthur Asuncion, Padhraic Smyth, and Max Welling. *Asynchronous distributed learning of topic models* , In Advances in Neural Information Processing Systems 21 (NIPS 2008), pages 81–88, Vancouver, British Columbia, Canada, 2008.
[10]. Ricardo Baeza-Yates, Carlos Castillo, Flavio Junqueira, VassilisPlachouras, and FabrizioSilvestri. *Challenges on distributed web retrieval*, In Proceedings of the IEEE 23rd International Conference on Data Engineering (ICDE 2007), pages 6–20, Istanbul, Turkey, 2007.
[11]. *MapReduce Algorithm*.URL: https://www.tutorialspoint.com/map_reduce /map_reduce_algorithm.htm
[12]. NogaAlon, Yossi Matias, and Mario Szegedy. The Space Complexity of Approximatingthe Frequency Moments. Journal of Computer and System Sciences, 58(1):137–147, 1999. 30, 116
[13]. M.R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. Technical report, Digital Systems Research Center, 1998. URL http://www.eecs.harvard.edu/~michaelm/E210/ datastreams.pdf. 30
[14]. J Feigenbaum, S Kannan, A McGregor, S Suri, and J Zhang. On Graph Problems in a Semi-Streaming Model. Theoretical Computer Science, 348(2-3):207– 216, 2005. 30
[15]. Jon Feldman, S. Muthukrishnan, AnastasiosSidiropoulos, Clifford Stein, and ZoyaSvitkina. On the Complexity of Processing Massive, Unordered, Distributed Data. Arxiv, 2007. 30
[16]. Brian Babcock, ShivnathBabu, MayurDatar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In PODS '02: 21st Symposium on Principles of Database Systems, pages 1–30, New York, New York, USA, June 2002. ACM Press. ISBN 1581135076. 30
[17]. Leonardo Neumeyer, Bruce Robbins, A. Nair, and A. Kesari. S4: Distributed Stream Computing Platform. In ICDMW '10: 10th

International Conference on Data Mining Workshops, pages 170–177. IEEE, 2010. 15, 23, 31

[18]. Gul Agha. ACTORS: A Model of Concurrent Computation in Distributed Systems. MIT Press, December 1986. ISBN 0-262-01092-5. 23, 32, 118

[19]. Data Intensive Computing. URL: https://en.wikipedia.org/wiki/Data-intensive_computing#cite_note-1

[20]. Leonardo Neumeyer, Bruce Robbins, A. Nair, and A. Kesari. S4: Distributed Stream Computing Platform. In ICDMW '10: 10th International Conference on Data Mining Workshops, pages 170–177. IEEE, 2010. 15, 23, 31

[21]. MEster, H P Kriegel, J Sander, and X Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In KDD '96: 2nd International Conference on Knowledge Discovery and Data mining, volume 1996, pages 226– 231. AAAI Press, 1996.

[22]. MihaelAnkerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: Ordering points to identify the clustering structure. In SIGMOD '99: 25th ACM International Conference on Management of Data, SIGMOD '99, pages 49–60, New York, NY, USA, 1999. ACM. ISBN 1-58113-084-8.

[23]. Tamer Elsayed, Jimmy Lin, and Douglas W Oard. Pairwise document similarity in large collections with MapReduce. In HLT '08: 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies, pages 265–268. Association for Computational Linguistics, June 2008.

[24]. RaresVernica, Michael J. Carey, and Chen Li. Efficient parallel set-similarity joins using MapReduce. In SIGMOD '10: 36th International Conference on Management of Data, pages 495–506, New York, New York, USA, 2010. ACM Press. ISBN 9781450300322.

[25]. Gianmarco De FrancisciMorales .*Big Data and theWeb: Algorithms forData Intensive Scalable Computing*, IMT Institute for Advanced Studies Lucca, Italy2012.