IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 2, February 2016

Energy Efficient Scheduling of Map Reduce for Evolving Big Data Applications

Mrs.P.Sheela Rani[1], S.Shalini[2], J.Rukmani@keerthika[3], A.Shanthini[4]

Assistant Professor, Department of Information Technology, Panimalar Institute of Technology, Chennai, India[1]

Student, Department of Information Technology, Panimalar Institute of Technology, Chennai, India[2,3,4]

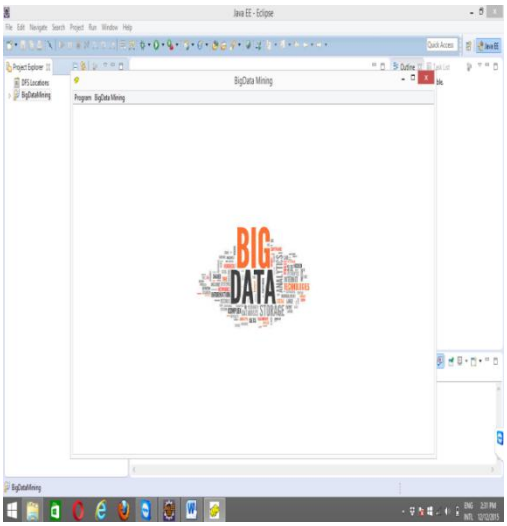
Abstract: In recent years the data mining applications become stale and obsolete over time. Energy wastage is the major problem more of the IT firms. More workload and more computational will increase high energy cost. Incremental processing is a promising approach to refreshing mining results. It utilizes previously saved states to avoid the expense of re-computation from scratch. In this paper, we propose Energy Map Reduce Scheduling Algorithm, a novel incremental processing extension to Map Reduce, the most widely used framework for mining big data. Map reduce is a programming model for processing and generating large amount of data in parallel time. In this paper, EMRSA is algorithm provide more energy and less maps. Priority based scheduling is a task will allocate the schedules based on necessary and utilization of the Jobs. For reducing the maps, it will reduce the system work so easily energy has improved. Final results show the experimental comparison of the different algorithms involved in the paper.

Keywords: BigData, EMRSA, MapReduce, incremental processing.

I.INTRODUCTION

Nowadays vast quantity of digital data is being accumulated in several important areas, including e-commerce, social network, finance,banking, health care, education, and environment. It's become progressively popular to mine such massive data so as to achieve insights to assist business selections or to produce higher customized, higher quality services. In recent years, a large variety of computing frameworks [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] are developed for large information analysis. Among these frameworks, mapreduce [1] (with its ASCII text file implementations, like Hadoop) is that the most generally utilized in production because of its simplicity, generality, and maturity. We tend to focus on up map reduce during this paper.It is now implemented with the bigdata applications.
Big data is constantly evolving. Due to the advent of new technologies, devices, and communication means like social networking sites, the amount of data produced by mankind is growing rapidly every year. Though all this information produced is meaningful and can be useful when processed, it is being neglectedBig data means really a big data, it is a collection of large datasets that cannot be processed using traditional computing techniques. Big data is not merely a data, rather it has become a complete subject, which involves various tools, techniques and frameworks.As new data and updates are being collected, the input data of a big data mining algorithm will gradually change, and the computed results will become stale and obsolete over time.

Incremental processing is a promising approach to refreshing mining results. It utilizes previously saved states to avoid the expense of re-computation from scratch. In this paper, we propose Energy Map Reduce Scheduling Algorithm, a novel incremental processing extension to Map Reduce, the most widely used framework for mining big data. MapReduce to support incremental processing. However, it has two main limitations.



First, Incoop supports only task-level incremental processing. . That is, it saves and reuses states at the coarseness of individual Map and reduce tasks. every task generally processes an oversized number of key-valuepairs (kvpairs). If Incoop detects any data changes within the input of a task, it'll rerun the whole task. whereas this approach simply leverages existing MapReduce options for state savings, it should incur a large quantity of redundant computation if solely a small fraction of kv-pairs have modified during a task. Second, Incoop supports solely one-step computation, whereas important mining algorithms, like PageRank, need iterative computation. Incoop would treat every iteration as a separate MapReduce job. However, a small number of input datachanges could gradually propagate to affect a large portion of intermediate states when variety of

DOI 10.17148/IJARCCE.2016.5213

IJARCCE

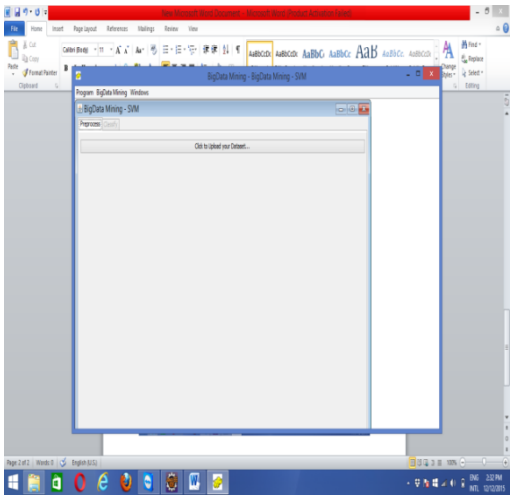
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 2, February 2016

iterations, leading to expensive global re-computation after. we tend to propose i2MapReduce, an extension to MapReduce that supports fine-grain incremental processing for each onestep and iterative computation.

II.PROPOSED METHODOLOGY

In our current situation energy wastage is the major problem more of the IT firms. More workload and more computational will increase high energy cost. Main aim of ours, to reduce the energy cost from efficient Map reducing concepts. To optimize the mining results, we evaluate MapReduce using a one-step algorithm and three iterative algorithms with diverse computation characteristics for efficient mining also improve the energy. In this paper we also include the algorithm for incremental processing approach named as Energy map reduce scheduling algorithm .EMRSA is algorithm provide more energy and less maps. Priority based scheduling is a task will allocate the schedules based on necessary and utilization of the Jobs. For reducing the maps, it will reduce the system work so easily energy has improve. Final results shows the experimental comaprison of the different algorithms involved in the paper.



III.RELATED WORK

(1) ENERGY MAP REDUCE SCHEDULING ALGORITHM

Algorithm 1 EMRSA-X

1: Create an empty priority queue $\mathcal{Q}^m$
2: Create an empty priority queue $\mathcal{Q}^r$
3: **for all** $j \in \mathcal{A}$ **do**
4: $ecr_j^m = \min_{\forall i \in \mathcal{M}} \frac{e_{ij}}{p_{ij}}$, for EMRSA-I; or
$ecr_j^m = \frac{\sum_{\forall i \in \mathcal{M}} \frac{e_{ij}}{p_{ij}}}{M}$, for EMRSA-II
5: $\mathcal{Q}^m$.enqueue(j, ecr_j^m)
6: **for all** $j \in \mathcal{B}$ **do**
7: $ecr_j^r = \min_{\forall i \in \mathcal{R}} \frac{e_{ij}}{p_{ij}}$, for EMRSA-I; or
$ecr_j^r = \frac{\sum_{\forall i \in \mathcal{R}} \frac{e_{ij}}{p_{ij}}}{R}$, for EMRSA-II
8: $\mathcal{Q}^r$.enqueue(j, ecr_j^r)
9: $D^m \leftarrow \infty$; $D^r \leftarrow \infty$
10: **while** $\mathcal{Q}^m$ is not empty **and** $\mathcal{Q}^r$ is not empty **do**
11: $j^m = \mathcal{Q}^m$.extractMin()
12: $j^r = \mathcal{Q}^r$.extractMin()
13: $f = \frac{\sum_{\forall i \in \mathcal{M}} p_{ij^m}}{\sum_{\forall i \in \mathcal{R}} p_{ij^r}}$
14: $\mathcal{T}^m$: sorted unassigned map tasks $i \in \mathcal{M}$ based on p_{ij^m}
15: $\mathcal{T}^r$: sorted unassigned reduce tasks $i \in \mathcal{R}$ based on p_{ij^r}
16: **if** $\mathcal{T}^m = \emptyset$ **and** $\mathcal{T}^r = \emptyset$ **then break**
17: ASSIGN-LARGE()
18: ASSIGN-SMALL()
19: **if** $D^m = \infty$ **then**
20: $D^m = D - p^r$
21: $D^r = p^r$
22: **if** $\mathcal{T}^m \neq \emptyset$ **or** $\mathcal{T}^r \neq \emptyset$ **then**
23: No feasible schedule
24: **return**
25: **Output:** X, Y

This paper involves the input files with the .arff extension, that is, artributte relation file format. An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files have two distinct sections. The first section is the Header information, which is followed the Data information.The Header of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types.

Hadoop plugin is implemented in the eclipse environment. Hadoop is the flexible and available architecture for large scale computation on data processing on a netwrk of commodity hardware. Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Here in this paper we are implementing hadoop plugin by including tha jar files in eclipse and which creates a virtual memory of 1GB.

IV.CLASSIFICATIONS

In this paper we show the efficient methods of classifying the evaluvated results by using the following two classification methods:

(i) Support Vector Machine(SVM)

(ii) Naive Bayesian.

(2.1) Support Vector Machine(SVM):

A support vector machine is a Classification method.Supervised algorithm used for:

- Classification and Regression (binary and multi-class problem)
- anomalie detection (one class problem)

An SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier.

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.

The support vector machine has been developed as robust tool for classification and regression in noisy, complex domains. The two key features of support vector machines are generalization theory, which leads to a principled way to choose an hypothesis; and, kernel functions, which introduce non-linearity in the hypothesis space without explicitly requiring a non-linear algorithm.

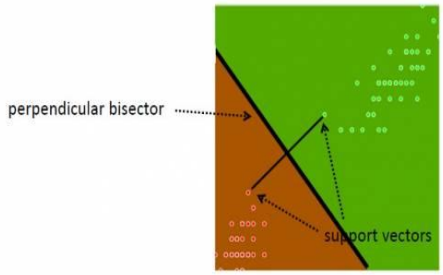
DOI 10.17148/IJARCCE.2016.5213

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 2, February 2016

(2.1.1) Support Vectors



The black line that separate the two cloud of class is right down the middle of a channel. The seperations is in 2D,aline, in 3D a plane, in four or more dimension an a hyper plane. Mathematically, the separation can be found by taking the two critical members, one for each class. This points are called support vectors.These are the critical points (members) that define the channel.The separation is then the perpendicular bisector of the line joining these two support vectors.That's the idea of support vector machine.

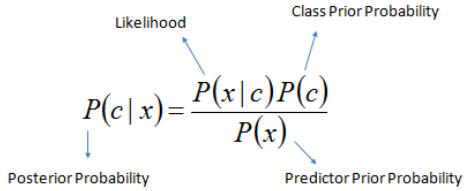
The SVM does not fall into the class of 'just another algorithm' as it is based on firm statistical and mathematical foundations concerning generalisation and optimisation theory. Moreover, it has been shown to outperform existing techniques on a wide variety of real world problems. SVMs will not solve all of your problems, but as kernel methods and maximum margin methods are further improved and taken up by the data mining community they will become an essential tool in any data miner's toolkit.

(2.2) Naive Bayesian

The Naive Bayesian classifier is based on Bayes' theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

Algorithm:

Bayes theorem provides a way of calculating the posterior probability, P(c|x), from P(c), P(x), and P(x|c). Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.

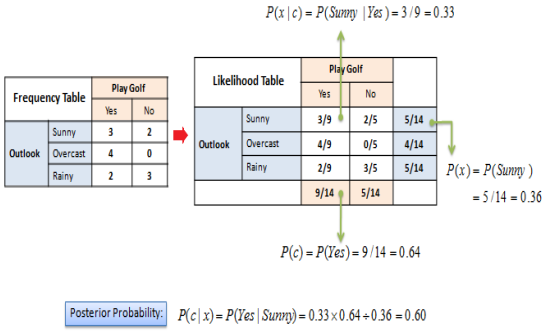


$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

- P(c|x) is the posterior probability of class (target) given predictor (attribute).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor.

Example:

The posterior probability can be calculated by first, constructing a frequency table for each attribute against the target. Then, transforming the frequency tables to likelihood tables and finally use the Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.



The zero-frequency problem

Add 1 to the count for every attribute value-class combination (Laplace estimator) when an attribute value (Outlook=Overcast) doesn't occur with every class value (Play Golf=no).

Numerical Predictors

Numerical variables need to be transformed to their categorical counterparts (binning) before constructing their frequency tables. The other option we have is using the distribution of the numerical variable to have a good guess of the frequency. For example, one common practice is to assume normal distributions for numerical variables.

The probability density function for the normal distribution is defined by two parameters (mean and standard deviation).

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \text{Mean}$$

$$\sigma = \left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2\right]^{0.5} \qquad \text{Standard deviation}$$

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \qquad \text{Normal distribution}$$

Example:

		Humidity									*Mean*	*StDev*
Play Golf	yes	86	96	80	65	70	80	70	90	75	79.1	10.2
	no	85	90	70	95	91					86.2	9.7

$$P(\text{humidity} = 74 \mid \text{play} = \text{yes}) = \frac{1}{\sqrt{2\pi}(10.2)} e^{\frac{-(74-79.1)^2}{2(10.2)^2}} = 0.0344$$

$$P(\text{humidity} = 74 \mid \text{play} = \text{no}) = \frac{1}{\sqrt{2\pi}(9.7)} e^{\frac{-(74-86.2)^2}{2(9.7)^2}} = 0.0187$$

DOI 10.17148/IJARCCE.2016.5213

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 2, February 2016

V.SYSTEM ARCHITECTURE

1. MAP REDUCE TASK

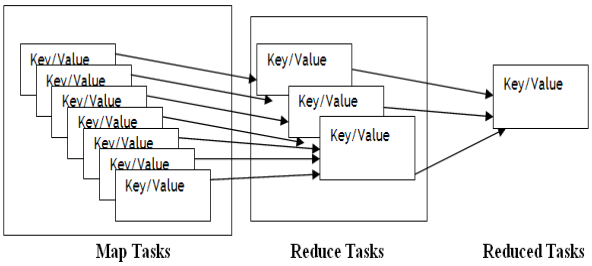


Fig. Map Reducing

MapReduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. MapReduce is the heart of Hadoop®. It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

2.SCHEDULING

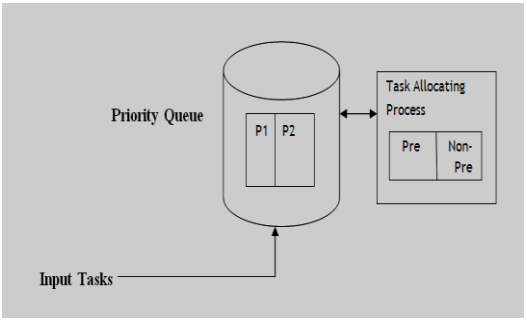


Fig. scheduling

Process scheduling is an essential part of a Multiprogramming operating system. Such operating systems allow more than one process to be loaded into the executable memory at a time and loaded process shares the CPU using time multiplexing. Priority Scheduling. The basic idea is straightforward: each process is assigned a priority, and priority is allowed to run. Equal-Priorityprocesses are scheduled in FCFS order. The shortest-Job-First (SJF)algorithm is a special case of general priority scheduling algorithm.

VI.PREVIOUS STUDY

Resilient dis-tributed datasets: A fault-tolerant abstraction for. in-memory cluster computing

Matei Zaharia Mosharaf Chowdhury Tathagata Das

We present Resilient Distributed Datasets (RDDs), a distributed memory abstraction that allows programmers to perform in-memory computations on large clusters while retaining the fault tolerance of data flow models like MapReduce. RDDs are motivated by two types of applications that current data flow systems handle inefficiently: iterative algorithms, which are common in graph applications and machine learning, and interactive data mining tools. In both cases, keeping data in memory can improve performance by an order of magnitude. To achieve fault tolerance efficiently, RDDs provide a highly restricted form of shared memory: they are read-only datasets that can only be constructed through bulk operations on other RDDs. Our implementation of RDDs can outperform Hadoop by 20× for iterative jobs and can be used interactively to search a 1 TB dataset with latencies of 5–7 seconds.

Drawbacks:

- Memory Sharing is more challenging one in this project.
- More Computational Cost.

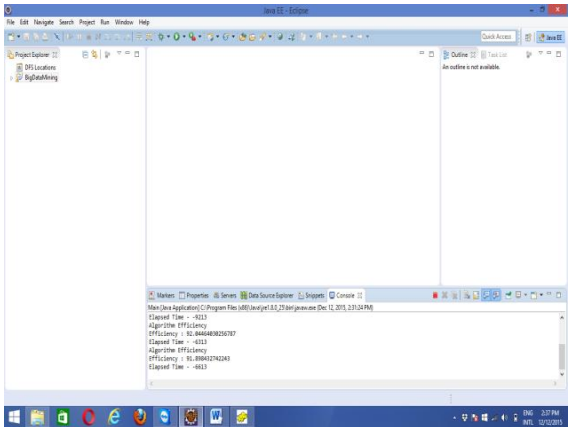
REX: Recursive, Delta-Based Data-Centric Computation
Svilen R. Mihaylov Zachary G. Ives Sudipto Guha

In today's Web and social network environments, query workloads include ad hoc and OLAP queries, as well as iterative algorithms that analyze data relationships (e.g., link analysis, clustering, learning). Modern DBMSs support ad hoc and OLAP queries, but most are not robust enough to scale to large clusters. Conversely, "cloud" platforms like MapReduce execute chains of batch tasks across clusters in a fault tolerant way, but have too much overhead to support ad hoc queries. Moreover, both classes of platform incur significant overhead in executing iterative data analysis algorithms. Most such iterative algorithms repeatedly refine portions of their answers, until some convergence criterion is reachedWe seek to unify the strengths of both styles of platforms, with a focus on supporting iterative computations in which changes, in the form of deltas, are propagated from iteration to iteration, and state is efficiently updated in an extensible way.We experimentally validate our techniques, and show speedups over the competing methods ranging from 2.5 to nearly 100 time .

Drawbacks:

- User-defined functions are also typically harder to write for DBMSs than for cloud platforms.
- High redundant of data
- High amount of I/O Overhead

VII.SOFTWARE IMPLEMENTATION



DOI 10.17148/IJARCCE.2016.5213

IJARCCE

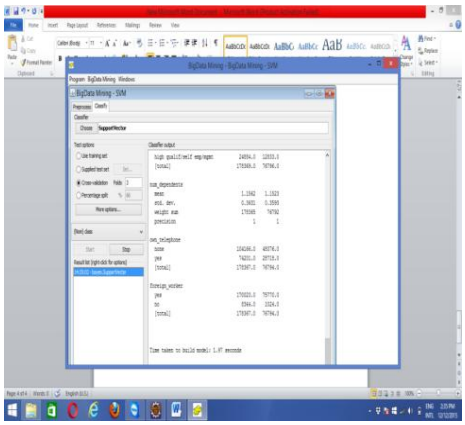
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 2, February 2016

Hadoop is a powerful framework that allows for automatic parallelezation of computing task. Unfortunately programming for it poses certain challenges, namely it is really hard to understand and debug Hadoop programs. One way to easy things a little bit is to have a simplified version of the hadoop cluster that could run locally on the developer's machine. This tutorial describes how to set-up such cluster on the computer running Microsoft Windows, also it describes how to integrate this cluster with the Eclipse development environment. Eclipse is a prime environment for Java development.

SETTING UP HADOOP PLUGIN:

1. Open another explorer window, either through "My Computer" icon or by using the "Start -> Run" menu. Navigate to your Eclipse installation and then open the "plugin" folder of your Eclipse installation.
2. Copy the file "hadoop-0.19.1-eclipse-plugin.jar, from the Hadoop eclipse plugin folder to the Eclipse plugins folder.
3. Close both explorer windows
4. Start Eclipse.
5. Click on the open perspective icon , which is usually located in the upper-right corner the eclipse application. Then select Other from the menu.
6. Select Map/Reduce from the list of perspectives and press "OK" button.
7. Now that the we installed and configured hadoop cluster and eclipse plugin i's a time to test the setup by running a simple project.



VIII.CONCLUSION

We have described support vector machine and naïve Bayesian classification methods for effective data analysis results and a set of efficient techniques for incremental iterative processing computation. Real time experiments will show that EMRSA and the described classification methods significantly reduce the run time for refreshing big data mining results compared to re-computation on both plain and iterative MapReduce thereby reduces the workload of the system which results in the efficient and reliable energy usage.

REFERENCES

[1] S. Lloyd, "Least squares quantization in PCM," IEEE Trans. Inform. Theory., vol. 28, no. 2, pp. 129–137, Mar. 1982.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in Proc. 20th Int. Conf. Very Large Data Bases, 1994, pp. 487–499.

[3]S. Brin, and L. Page, "The anatomy of a large-scale hypertextual web search engine," Comput. Netw. ISDN Syst., vol. 30, no. 1–7, pp. 107–117, Apr. 1998.

[4] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in Proc. 6th Conf. Symp. Opear. Syst. Des. Implementation, 2004, p. 10.

[5] R. Power and J. Li, "Piccolo: Building fast, distributed programs with partitioned tables," in Proc. 9th USENIX Conf. Oper. Syst. Des. Implementation, 2010, pp. 1–14.

[6] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 135–146.

[7] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, "Haloop: Efficient iterative data processing on large clusters," in Proc. VLDB Endowment, 2010, vol. 3, no. 1–2, pp. 285–296.

[8] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, "Twister: A runtime for iterative mapreduce," in Proc. 19th ACM Symp. High Performance Distributed Comput., 2010, pp. 810–818.

[9] D. Peng and F. Dabek, "Large-scale incremental processing using distributed transactions and notifications," in Proc. 9th USENIX Conf. Oper. Syst. Des. Implementation, 2010, pp. 1–15.

[10] D. Logothetis, C. Olston, B. Reed, K. C. Webb, and K. Yocum, "Stateful bulk processing for incremental analytics," in Proc. 1st ACM Symp. Cloud Comput., 2010, pp. 51–62.

[11] J. Cho and H. Garcia-Molina, "The evolution of the web and implications for an incremental crawler," in Proc. 26th Int. Conf. Very Large Data Bases, 2000, pp. 200–209.

[12] C. Olston and M. Najork, "Web crawling," Found. Trends Inform. Retrieval, vol. 4, no. 3, pp. 175–246, 2010.

[13] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar, and R. Pasquin, "Incoop: Mapreduce for incremental computations," in Proc. 2nd ACM Symp. Cloud Comput., 2011, pp. 7:1–7:14.

[14] Y. Zhang, Q. Gao, L. Gao, and C. Wang, "Priter: A distributed framework for prioritized iterative computations," in Proc. 2nd ACM Symp. Cloud Comput., 2011, pp. 13:1–13:14.

[15] T. J€org, R. Parvizi, H. Yong, and S. Dessloch, "Incremental recomputations in mapreduce," in Proc. 3rd Int. Workshop Cloud Data Manage., 2011, pp. 7–14.

[16] Y. Zhang, Q. Gao, L. Gao, and C. Wang, "imapreduce: A distributed computing framework for iterative computation," J. Grid Comput., vol. 10, no. 1, pp. 47–68, 2012.

[17] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for, in-memory cluster computing," in Proc. 9th USENIX Conf. Netw. Syst. Des. Implementation, 2012, p. 2.

[18] S. R. Mihaylov, Z. G. Ives, and S. Guha, "Rex: Recursive, deltabased data-centric computation," in Proc. VLDB Endowment, 2012, vol. 5, no. 11, pp. 1280–1291.

[19] Y.Zhang,Q.Gao,L.Gao,andC.Wang,"Acceleratelarge-scaleiterative computation through asynchronous accumulative updates," inProc.3rdWorkshopSci.CloudComput.Date,2012,pp.13–22.

[20] C. Yan, X. Yang, Z. Yu, M. Li, and X. Li, "IncMR: Incremental data processing based on mapreduce," in Proc. IEEE 5th Int. Conf. Cloud Comput., 2012, pp.pp. 534–541.

[21] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: A framework for machine learning and data mining in the cloud," in Proc. VLDB Endowment, 2012, vol. 5, no. 8, pp. 716–727.

[22] S. Ewen, K. Tzoumas, M. Kaufmann, and V. Markl, "Spinning fast iterative data flows," in Proc. VLDB Endowment, 2012, vol. 5, no. 11, pp. 1268–1279.

[23] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi, "Naiad: A timely dataflow system," in Proc. 24th ACM Symp. Oper. Syst. Principles, 2013, pp. 439–455.

[24] U. Kang, C. Tsourakakis, and C. Faloutsos, "Pegasus: A peta-scale graph mining system implementation and observations," in Proc. IEEE Int. Conf. Data Mining, 2009, pp. 229–238.

[25] Y. Zhang, S. Chen, Q. Wang, and G. Yu, "i2mapreduce: Incremental mapreduce for mining evolving big data," CoRR, vol. abs/ 1501.04854, 201.