ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

I2MapReduce: Fine-Grain Incremental Processing in Big Data Mining

Kulkarni Mugdha A, Prof. Shaikh Imran R.

Student of M.E (Computer Engineering), S.N.D. College of Engineering and Research Centre, Yeola, Pune University Maharashtra, India

Assistant Professor, Dept. of C.S., S.N.D. College of Engineering and Research Centre, Yeola, Pune University, Maharashtra, India

ABSTRACT: I2mapreduce: fine-grain incremental processing in big data mining is the extension of Map reduces technique it improves the stale and obsolete data mining application results as the new data and updates are arrives. Incremental processing gives refreshing mining results I2MapReduce consider its own merits (i) It promotes a important services to execute value-key pair computational level processing instead of detail level re-calculation, (ii) It assists single-step computation along with extra cultivated continual computation, and it is mostly implemented in data mining application, and also (iii) By Incorporating the set of novel techniques, can decrement the I/O overhead for collecting the conserved fine-drawn computation states.I2MAPREDUCE:FINE-GRAIN INCREMENTAL PROCESSING IN BIG DATA MINING holds the rare incremental processing for both single step and continual computation. By using Map reduce based graph store algorithm. The Preliminary outcome at Amazon EC2 represents the powerful improvement in performance of I2MapReduce, as compared to both transparent and continual MapReduce, which perform the re-computation.

KEYWORDS: Iterative processing, MapReduce, Iterative computation, large-scale data processing

I.INTRODUCTION

For every organization data is important for gaining the knowledge, annotation and research the enormous number of collection of data is present in multiple dimensions such as e-business, social network, financial sector, medical sector, education, and environment. Healthcare contain huge amount of data which is nothing but big data solution for potential impact. Mining of this big data in healthcare is very important in order to provide better personalized, higher quality services to the patient. Well famous map reduce programming structural model as well as a file system supports this model is called as Hadoop Distributed File System (HDFS) processes the data and also help to analyze unstructured data into structured data. Data sources are having high dimensional data of patient, provider can collect all the data and store it into structured database. The admin can access the provided data. Data allocated from provider only. If n size given in relation then there are 2 raise to n cuboids and this cuboids computed in the cube materialization using algorithm [2] which is able to support the feature inMapReduce for efficient cube computation. MapReduce programming model processes large volumes of data in parallel manner by dividing the work into a set of autonomous tasks. MapReduce based algorithm uses that introduce efficient cube computation [5] and identifies cube sets/groups on non-algebraic measures. Complexity is depends on things.

II. LITERATURE SURVEY

There has been number of studies on MapReduce techniques for learning Incremental Iterative MapReducing, Parallel Data Processing with MapReducing, MapReducing with data processing on large clusters, MapReducing computational model, Big Data Mining using Map Reduce, MapReducing in cloud computing environment, New MapReduce Scheduling Technique and Runtime MapReducing.
I2MapReduce: Incremental Iterative MapReduce Iterative computations like PageRank are always done by Cloud intelligence Applications where database and dataset changes are constant, for example Web-graph. For well-organized

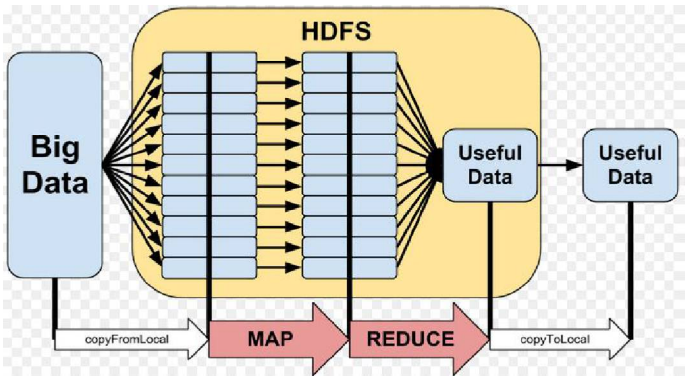
iterative computations are extend by MapReduce and these is already done in past studies, it is too exorbitant to design an entirely new big scale MapReduce iterative job to accommodate new changes to the underlyingData sets within a certain time limit. The proposed method in this paper is, i2MapReduce to assist incremental iterative computation. In multiple cases it is distinguish that, the present iteratively converged state is totally close to past converged state and these effect changes only a slight fraction of the data sets.

Parallel Data Processing with MapReduce: A Survey a renowned equidistant data processing mechanism of MapReduce is remarkably achieving a strength from both internet industry and education industry, because the volume of data to analyze is spreading very quickly. For heavy and immense data observation and its performance analysis MapReduce is applicable in multiple domains, there are still formal discussion on a particular parameters such as its performance analysis, per node efficiency, and easy abstraction.

MapReduce: Simplified Data Processing onLarge Clusters A corresponding application for data set processing and provoking a big data sets is implemented by MapReduce Programming Model. Users define a map function that processes a value/key pair to promote a set of midway value/key pairs, and a lower functionality that unify all intermediate keys connect with the same intermediate values.. Various real time environmental functions are expressible in this model. A Model of Computation for MapReduceIn modern technology the MapReduceframework has become well-known platform for parallel computing and it is extensively applicable for data processing on terabyte and petabyte scales. Multi -National Companies such as Yahoo!, Google, Amazon, and Facebook, and currently acquired by different universities, it supports easy parallelization of data intensive computations over various machines. One vital aspect of MapReduce is that, it is differentiate from previous models of parallel computation and that interleaves concurrent computation as well as logical computation. Big Data Mining using Map Reduce Largescale data application such as Hadoop file system database has rapidly growing day by day.it is very exorbitant to manage, apprehend and bring out the processing data using software tools which is already in existing. Large scale data sets are big in size, miscellaneous and dispense. For example Weather Forecasting, Electricity Demand Supply, social media and so on. With increasing size of data in data warehouse it is expensive to perform data analysis. Data cube commonly abstracting and summarizing databases. It is way of structuring data in different n dimensions for analysis over some measure of interest. For data processing bigdata processing framework relay on cluster computers and parallel execution framework provided by Map-Reduce. Industry is being challenged to develop methods and techniques for affordable data processing on large datasets at optimum response times. The technical challenges in dealing with the increasing demand to handle vast quantities of data is daunting and on the rise. One of the recent processing models with a more efficient and intuitive solution to rapidly process large amount of data in parallel is called MapReduce. It is a framework defining a template approach of programming to perform large-scale data computation on clusters of machines in a cloud computing environment.

Matchmaking: A New MapReduce Scheduling Technique This paper introduced a new intensify scheduling technique of MapReduce to boost the locality of map tasks. For processing of big data sets a MapReduce scheduling is a robust platform. Good performance is simply reached by a MapReduce scheduler and keep away from unnecessary data transmission by upgrading the location of data (implementing tasks on nodes that hold data for input).

III. PROPOSED WORK



MapReduce, a novel incremental processing extension to MapReduce, themost widely used framework for mining big data. Compared with the state-of-the-art work on Incoop, i2MapReduce performs key-valuepair level incremental processing rather than task level re-computation, supports not only one-step computation but also moresophisticated iterative computation, which is widely used in data mining applications, and incorporates a set of novel techniques toreduce I/O overhead for accessing preserved fine-grain computation states. We evaluate i2MapReduce using a one-step algorithm andfour iterative algorithms with diverse computationcharacteristics. It show significant performanceimprovements of i2MapReduce compared to both plain and iterative MapReduce performing re-computation.We propose a generalpurpose MapReduce model for iterative computation and describe how to efficiently support this computationi2MapReduce must transfer the updatedstate kv-pairs to their corresponding prime Map task, which caches the independent structure kv-pairs in itslocal file system. Real-machine experiments showthat i2 MapReduce can significantly reduce the run timefor refreshing big data mining results compared to recomputationon both plain and iterative MapReduce.

A. Advantages

It performs key-value pair level incremental processing.It supports one-step computation and more sophisticated iterative computation. Performance is very high in Runtime. Consider two MapReduce jobs A and A0 performing the same computation on input data set D and D0, respectively. D0 ¼ D þ DD, where DD consists of the inserted and deleted input hK1; V 1is1. An update can be represented as a deletion followed by an insertion. Our goal is to recomputed only the Map and Reduce function call instances that are affected by DD. Incremental computation for Map is straightforward. We simply invoke the Map function for the inserted or deleted hK1; V 1is. Since the other input kv-pairs are not changed,their Map computation would remain the same. We now have computed the delta intermediate values, denoted DM,including inserted and deleted hK2; V2is.To perform incremental Reduce computation, we need to save the fine-grain states of job A, denoted M, which includes hK2; fV 2gis. We will recompute the Reduce function for each K2 in DM. The other K2 in M does not see anychanged intermediate values and therefore would generate the same final result. For a K2 in DM, typically only a subset of the list of V 2 have changed. Here, we retrieve the saved hK2; fV 2gi from M, and apply the inserted and/or deleted values from DM to obtain an updated Reduce input.We then re-compute the Reduce function on this input to generate the changed final results hK3; V 3 is.it is easy to see that results generated from this incremental computation are logically the same as the results from completely re-computing A0.

B.Incremental iterative processing

In this section, we present incremental processing techniques for iterative computation. Note that it is not sufficient to simply combine the above solutions for incremental one step processing and iterative computation. In the following, we discuss three aspects that we address in order to achieve an effective design.

C. Fault-Tolerance

Vanilla MapReduce reschedules the failed Map/Reduce task in case task failure is detected. However, the interdependency of prime Reduce tasks and prime Map tasks in i2MapReduce requires more complicated fault-tolerance solution. I2MapReduce checkpoints the prime Reduce task's output state data and MRBGraph file on HDFS in every iteration.Upon detecting a failure, i2MapReduce recovers by considering task dependencies in three cases. (i) In case a prime Map task fails, the master reschedules the Map task on the worker where its dependent Reduce task resides. The prime Map task reloads its structure data and resumes computation from its dependent state data (checkpoint). (ii) In case a prime Reduce task fails, the master reschedules the Reduce task on the worker where its dependent Map task resides. The prime Reduce task reloads its MRBGraph file (checkpoint) and resumes computation by re-collecting Map outputs. (iii) In case a worker fails, the master reschedules the interdependent prime Map task and prime Reduce task to a healthy worker together. The prime Map task and Reduce task resume computation based on the check pointed state data and MRBGraph file as introduced above.

D.Reducing Change Propagation

In incremental iterative computation, changes in the delta input may propagate to more and more kv-pairs as the computation iterates. For example, in PageRank, a change that affects a vertex in a web graph propagates to the neighbor vertices after an iteration, to the neighbors of the neighbors after two iterations, to the neighbors after three iterations, and so on. Due to this effect, incremental processing may become less effective after a number of iterations. To address this problem, i2MapReduce employs aChange propagation control technique, which is similar to the dynamic computation in Graph Lab [6]. It filters negligible changes of state kv below a given threshold. These filtered kv supposed to be very close to convergence. Onlythe state values that see changes greater than the thresholdare

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

emitted for next iteration.The changes for a state kv-pair are accumulated. It is possible a filtered kv-pair may later be emitted if its accumulated change is big enough .The observation behind this technique is that iterative computation often converges asymmetrically: Many state kv quickly converge in a few iterations, while the remaining state kv-pairs converge slowly over many iterations.

E.MRBG-Store

The MRBG-Store supports the preservation and retrieval offline-grain MRBGraph states for incremental processing. We see two main requirements on the MRBG Store must incrementally store the evolving MRBGraph. Consider a sequence that incrementally refresh the results of a big data mining algorithm. As input data evolves, the intermediate states in the MRBGraph will also evolve. It would be wasteful to store the entire each subsequent job. Instead, we would like and store only the updated part of the MRBGraph. Second, the MRGB efficient retrieval of preserved states of given Reduce instances. For incremental computation, i2MapReduce re instance associated with edge, as described in Section 3.3. For a changededge, it queries the MRGB preserved states of the in-edges of the associated K2, and merge the preserved states with the newly computed edge changes.

IV. ALGORITHMS

A.Query Algorithm in MRBG-Store

```
Input: queried key: k; the list of queried keys:L
Output: chunk k
1: if! read cache. Contains (k) then
2: gap =0, w =0
3: i=ks index in L That is, L i = k
4: while gap ¡ T and w + gap + length (L i) read cache:
Size do
5: w =w+gap + length (L i)
6: gap= pos(L i+1) pos(L i) - length (L i)
7: i =i + 1
8: end while
9: starting from posk, read w bytes into read cache
10: end if
11: return read cache. Getchunk(k)
```

B.Page ranking Algorithm

```
Map Phase Input: <i,Ni|Ri>
1: output<i,Ni>
2: for all j in Nido
3: Ri,j= Ri |Ni|
4: output<j,Ri,j>
5: end for
Reduce Phase input:<j,{Ri,j,Ni}>
6: Rj=dΣiRi,j+(1-d)
7: output<j,Nj|Rj>
```

C.K-Means Clustering Algorithm

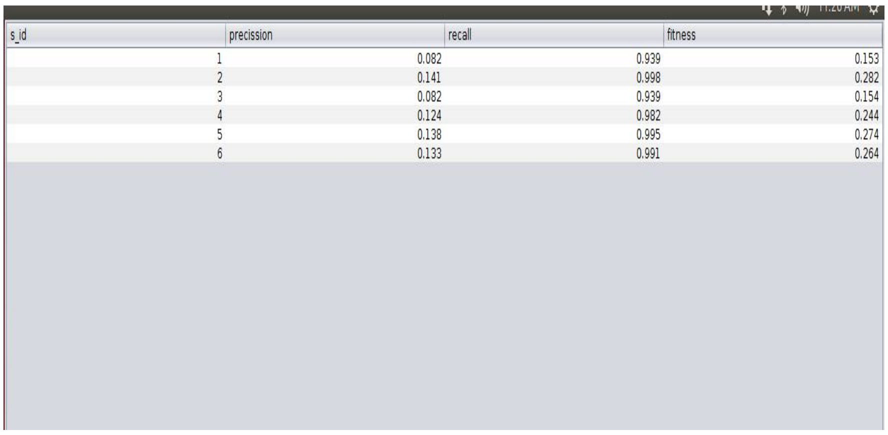
1. Clusters predefined.
2. Select k points at random as cluster centers.
3. Assign objects to their closest cluster center according to the Euclidean distance function.
4. Calculate the centroid or mean of all objects in each cluster. The data into k groups where k is
5. Repeat steps 2, 3 and 4 until the same points are assigned to each cluster in consecutive rounds

D.Algorithm: GIM-V in Map Reduce

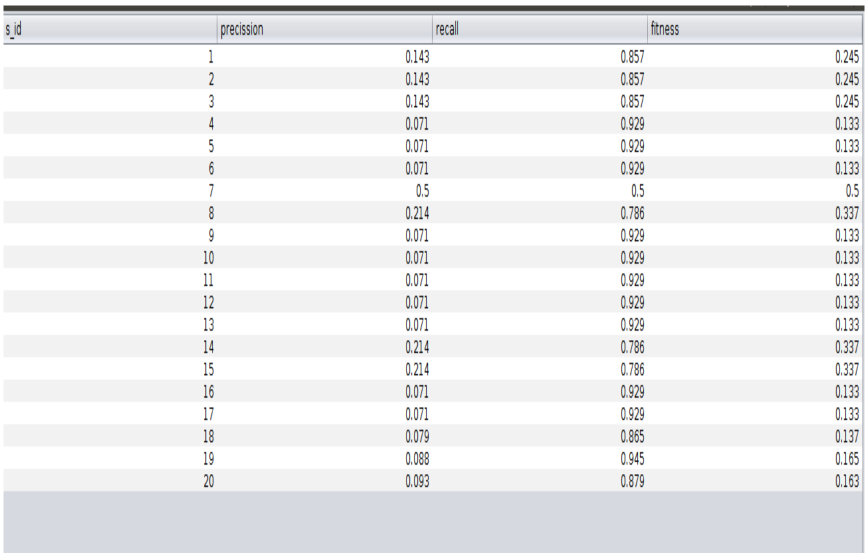
```
Map Phase 1 input: < (i,j),mi,j>or<j,vj>
1: if kv-pair is < (i,j),mi,j>then
2: output < (i,j),mi,j>
3:  else if kv-pair is <j,vj>then
```

```
4: for all i blocks in j’s row do
5: output < (i,j),vj>
6: end for
7: end if
Reduce Phase 1 input: < (i,j),{mi,j,vj}>
8: mvi,j=combine2(mi,j,vj)
9: output<i,mvi,j>,<j,vj>
Map Phase2: output all inputs
Reduce Phase2: input<i, {mvi,j,vi}>
10:v’i=combine All ({mvi,j})
11: vi=assign (vi,v’i)
12: output<i,vi>
```

V. RESULT ANALYSIS



s_id	precission	recall	fitness
1	0.082	0.939	0.153
2	0.141	0.998	0.282
3	0.082	0.939	0.154
4	0.124	0.982	0.244
5	0.138	0.995	0.274
6	0.133	0.991	0.264



s_id	precission	recall	fitness
1	0.143	0.857	0.245
2	0.143	0.857	0.245
3	0.143	0.857	0.245
4	0.071	0.929	0.133
5	0.071	0.929	0.133
6	0.071	0.929	0.133
7	0.5	0.5	0.5
8	0.214	0.786	0.337
9	0.071	0.929	0.133
10	0.071	0.929	0.133
11	0.071	0.929	0.133
12	0.071	0.929	0.133
13	0.071	0.929	0.133
14	0.214	0.786	0.337
15	0.214	0.786	0.337
16	0.071	0.929	0.133
17	0.071	0.929	0.133
18	0.079	0.865	0.137
19	0.088	0.945	0.165
20	0.093	0.879	0.163

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

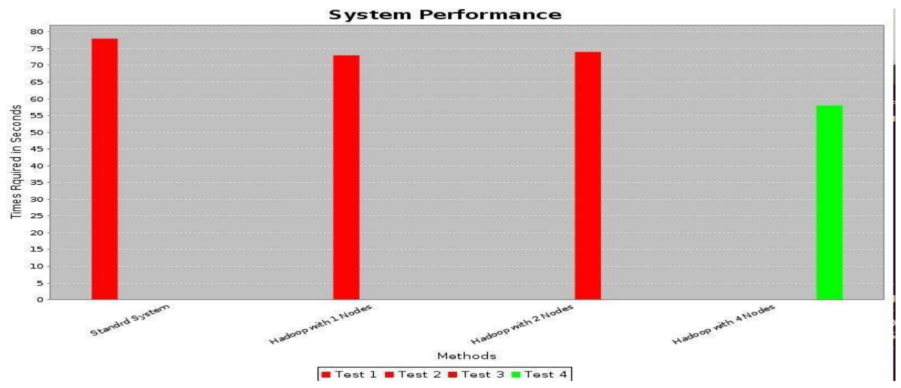
International Journal of Innovative Research in Computer and Communication Engineering

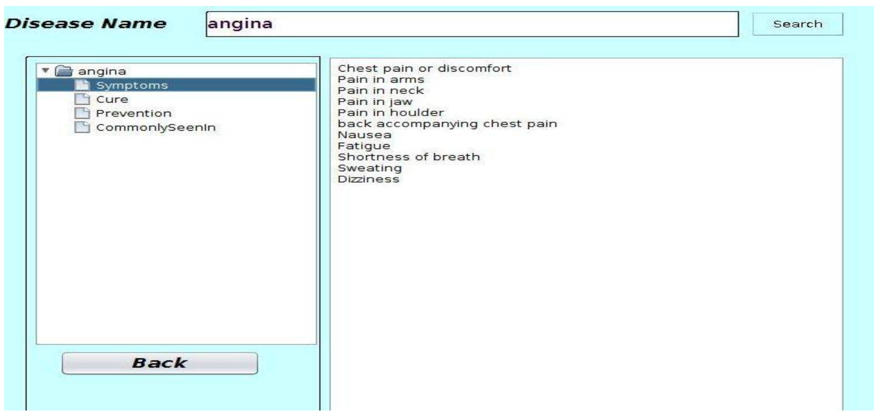
(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Precision & Recall

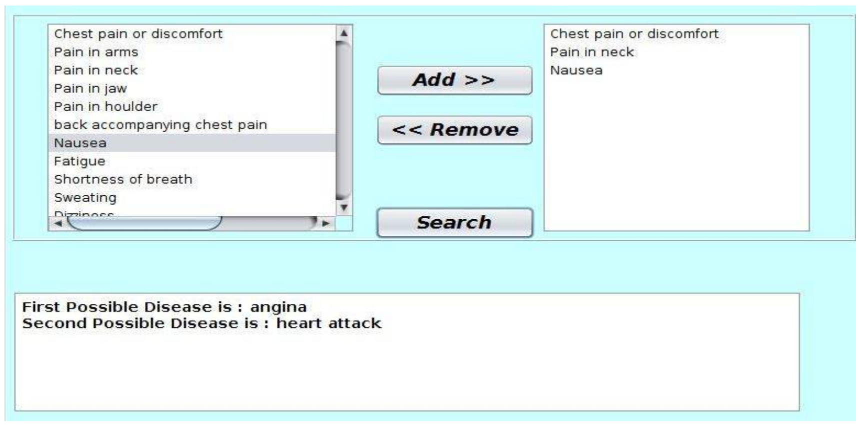
Precision and recall are calculated based on the number of record retrieved. Precision shows number of relevant record retrieved as per the user query out of total record in the database.Recall is calculated number of irrelevant record retrieved as per the user query out of total record.The precision and recall is calculated for both disease as well as symptoms.This analysis is help to calculate the accuracy.





Disease Name: angina

Chest pain or discomfort
Pain in arms
Pain in neck
Pain in jaw
Pain in houlder
back accompanying chest pain
Nausea
Fatigue
Shortness of breath
Sweating
Dizziness



Chest pain or discomfort
Pain in arms
Pain in neck
Pain in jaw
Pain in houlder
back accompanying chest pain
Nausea
Fatigue
Shortness of breath
Sweating
Dizziness

Chest pain or discomfort
Pain in neck
Nausea

First Possible Disease is : angina
Second Possible Disease is : heart attack

VI. MATHEMATICAL MODEL

I= I1,I2,I3,I4
Where,
I1=Username
I2=Password
I3=Keywords
I4=Dataset
Intermediate Output Set
E= E1, E2
Where,
E1=Authorized User
E2=Homogeneous Data
E3=Heterogeneous Data
E4= Anonymous Data
Final Output Set
D= D1

VII. CONCLUSION

I have described I2MapReduce- Fine Grain Incremental Processing based on MapReduce framework. That implements the key-pair level, which is rare incremental process to decline the number of re-computation and MRBG-Store to carry the most elegant quires to recapture the rare states for cumulative processing and to retain the rare states in MRBGraph. This framework combines, a regular purpose continual model, a rare cumulative engine, and a competent techniques for fine-grain incremental continual computation. I2MapReduce can powerfully decrement the run time for reviving a large-scale data mining results in Real-machine experiments that compared to re-computation on both transparent and iterative MapReduce.

ACKNOWLEDGMENTS

I would like to give wholehearted thanks to my special mentor as well as advisor Prof. I.R Shaikh, who so generously guiding me to complete this project. First and the most important, I would like to give thanks to God Almighty for His mercy and beautiful blessings in my life. Even in the hardest times, He has always been there to guide and comfort me and to make sure that I never stray too far from His narrow way. I would like to thank my family for their unconditional love and support. I would especially thank my father and my mother. I express my sincere gratitude to Prof. S. R. Durugkar, Head of Department (Computer), Prof.G.R. Sonawane, (PG Co-ordinator).I also express my deep gratitude to all teaching and non-teaching staff of my computer department for providing me help whenever it was required.

REFERENCES

1 J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” in Proc. 6th Conf. Symp. Opear. Syst. Des. Implementation, 2004, p. 10.
2 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.
3 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.
4 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.
5 S. R. Mihaylov, Z. G. Ives, and S. Guha, “Rex: Recursive, delta based data-centric computation,” in Proc. VLDB Endowment, 2012, vol. 5, no. 11, pp. 1280–1291.
6 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.
7 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.
8 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.

9 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 Compute. 2010, pp. 810–818.
10 Y. Zhang, Q. Gao, L. Gao, and C. Wang, "imapreduce: A distributed computing framework for iterative computation," J. Grid Compute., vol. 10, no. 1, pp. 47–68, 2012.
11 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.
12 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.
13 D. Logothetis, C. Olston, B. Reed, K. C. Web, and K. Yocum,"Stateful bulk processing for incremental analytics," in Proc. 1st ACM Symp. Cloud Comput., 2010, pp. 51–62.
14 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.
15 P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar, and R. Pasquin, "Incoop: Mapreduce for incremental csssssomputations," in Proc. 2nd ACM Symp. Cloud compute. 2011, pp. 7:1–7:14.