IJARCCE

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

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

Intersection Analytics Systems Provenance in Emergence of Data Lakes from Dataware House

Akash Suryawanshi[1], Pratik Patil[2]

Computer Engineering B.V.D.U.C.O.E., Pune, India[1]

Information Technology B.V.D.U.C.O.E., Pune, India[2]

Abstract: The volumes of information in Big Data, their assortment and unstructured nature, have had analysts looking past the information distribution center. The information distribution center, among different highlights, requires mapping information to a composition upon ingest, an approach seen as unbendable for the huge assortment of Big Data. The Data Lake is rising as a substitute answer for putting away information of broadly dissimilar sorts and scales. Intended for high adaptability, the Data Lake takes after a blueprint on-read theory and information changes are thought to be performed inside the Data Lake. Amid its lifecycle in a Data Lake, an information item may experience various changes performed by any number of Huge Data preparing motors prompting inquiries of traceability. In this paper we contend that provenance adds to less demanding information administration and traceability inside a Data Lake framework. We talk about the difficulties in provenance reconciliation in a Data Lake and propose a reference design to overcome the challenges. We assess our design through a model execution constructed utilizing our disseminated provenance accumulation apparatuses.

Keywords: Big-Data, Dataware House.

I. INTRODUCTION

Industry, the scholarly community, and research alike are pondering the open doors that Big Data gets mining information from various hotspots for understanding, basic leadership, and prescient estimates. These sources (e.g., clickstream, sensor information, IoT gadgets, web-based social networking, server logs) are as often as possible both outer to an association and inward. Information from sources, for example, web-based social networking and sensors is produced persistently. Depending on the source, information can be organized, semi-organized, or unstructured. The customary arrangement, the information distribution center, is demonstrating unyielding and constrained as an information administration system in help of numerous investigation stages and information of various sources and sorts. The reaction to the furthest reaches of the information distribution center is the Data Lake. A component of the Data Lake is its pattern on-read (rather than outline on-compose which occurs at ingest time) where duties regarding a specific outline are conceded to time of utilization. Construction on-read proposes that information are ingested in a crude shape, at that point changed over to a specific construction as expected to complete examination. The Data Lake worldview recognizes that high throughput examination stages being used today are fluctuated, so needs to help various Big Data handling structures like Apache Hadoop1, Apache Spark2 and Apache Storm3. The vision of the Data Lake is one of information from various sources being dropped into the lake rapidly and effectively, with devices around the lake as assigned fishers of the lake, expectation on getting knowledge by the rich biological system of information inside the Information Lake. This more prominent adaptability of the Data Lake prompts rich accumulations of information from different sources. It, be that as it may, leaves more prominent reasonability loads in the hands of the lake overseers. The Data Lake can without much of a stretch turn into a "dump everything" put because of nonappearance of any implemented construction, at times alluded to in writing as an "information overwhelm". The Data Lake could effortlessly disregard the way that information things in a Data Lake can exist in distinctive phases of their life cycle. One information thing might be in crude organize after late age where another might be the fined consequence of examination by at least one of the investigation devices. The intricacies of information life cycle expand the requirement for legitimate traceability systems. In this paper the basic concentration of our consideration is on metadata and ancestry data through a information life cycle which are critical to great information openness and traceability.

Data provenance is the information about the activities, entities and people who involved in producing a data product. Data provenance is often represented in one of two standard provenance representations (i.e., OPM or PROV). Data provenance can help with management of a Data Lake by making it clear where an object is in its lifecycle. This information can ease the burden of transformation needed by analysis tools to operate on a dataset. For instance, how does a researcher know what datasets are available in the lake for Apache Spark analysis, and which can be available through a small amount of transformation? This information can be derived by hand by the lake administrator and stored to registry, but that approach runs counter to the ease with which new data can be added to the data lake. Another issue with the Data Lake is trust. Suppose a Data Lake is set up to organize data for the watershed basin of the Lower

IJARCCE

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

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

Mekong River in southeast Asia. The contributors are going to be from numerous countries through which the Mekong River passes. How does the Data Lake ensure that the uses of the data in the lake are proper and adhere to the terms of contribution?

How does a researcher, who uses the Data Lake for their research, prove that their research is done in a way that is consistent with the terms of contribution? Provenance contributes to these questions of use and trust. If the Data Lake framework can ensure that every data product's lineage and attribution are in place starting from the origin, critical traceability can be had. However, that is a challenging task because a data product in a Data Lake may go through different analytics systems such as Hadoop, Spark and Storm which do not produce provenance information by default. Even if there are provenance collection techniques for those systems, they may use their own ways of storing provenance or use different standards. Therefore generating integrated provenance traces across systems is difficult. In this paper we propose a reference architecture for provenance in a Data Lake based on a central provenance subsystem that stores and processes provenance events pumped into it from all connected systems. The reference architecture, which appeared in early work as a poster, is deepened here. A prototype implementation of the architecture using our distributed provenance collection tools shows that the proposed technique can be introduced into a Data Lake to capture integrated provenance without introducing much overhead.

The paper's three main contributions are: identification of the data management and traceability problems in a Data Lake that are solvable using provenance highlighting the challenges in capturing integrated provenance. Second, a reference architecture to overcome those challenges. Third, an evaluation of the viability of the proposed architecture using a prototype implementation with techniques that can be used to reduce the overhead of provenance capture.

II. DATA LAKE ARCHITECTURE

He general engineering of a Data Lake, appeared in Figure 1, contains three principle exercises: (1) Data ingest, (2) Data preparing or change and (3) Data investigation. A Data Lake may open up number of ingest APIs to bring information from diverse sources into the lake. As a rule, crude information is ingested into the Data Lake for later use by analysts for numerous reasons at various purposes of time. Movement in the Information Lake can be seen as information change: where information in the Data Lake is contribution to some undertaking, and yield is put away back to the Data Lake. Present day vast scale circulated Big Information preparing systems like Hadoop, Spark and Storm are the wellspring of such changes, particularly for Data Lakes actualized on HDFS. Systems like logical work process frameworks, for example, Kepler and inheritance contents may apply as well. An information item made as an yield from one change can be a contribution to another change which itself may create another as a result. At last when all handling advances are done, coming about information items are utilized for various types of examination reports what's more, expectations.

A. Part of Provenance

The Data Lake accomplishes expanded adaptability at the cost of decreased reasonability. In the exploration information condition, at the point when contrastingly organized information is ingested by various associations through one of numerous APIs, following progresses toward becoming an issue. Binded changes that persistently infer new information from existing information in the lake additionally convolute the administration. By what means can insignificant administration be included to the lake without negating the alluring advantage of straightforwardness of ingests? We place that this negligible administration is in the type of instruments to track sources of the information items, privileges of utilization and reasonableness of changes connected to them, and nature of information produced by the changes. Deliberately caught provenance can fulfill these necessities permitting, for example, answers to the accompanying two inquiries: 1.) Suppose touchy information are stored into a Data Lake; sociology overview information for example. Would data be able to provenance anticipate disgraceful spillage into inferred information? 2.) Repeating a Enormous Data change in a Data Lake is costly due to high asset and time utilization. Can live gushing provenance from tests distinguish issues at a very early stage in their execution?

B. Difficulties in Provenance Capture

Information in a Data Lake may experience number of changes performed utilizing diverse systems chose as indicated by the kind of information and application. For instance, in a HDFS based Data Lake, it is basic to utilize Storm or Start Streaming for gushing information and Hadoop MapReduce or, then again Spark for cluster information. Other inheritance frameworks and contents may be incorporated too. To accomplish traceability crosswise over changes, provenance caught from these frameworks must be coordinated, a test since many don't bolster provenance of course. Systems exist to gather provenance from Big Data handling systems like Hadoop and Spark. Be that as it may, most are coupled to a specific structure. In the event that the provenance gathering inside a Data Lake relies upon such framework particular strategies, provenance from all subsystems ought to be sewed together to make a more profound provenance follow. There are sewing procedures which bring all provenance follows into a typical model and after that coordinate them together.

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

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

C. Provenance Integration Across Systems

To address provenance integration, we propose a central provenance collection system to which all components within the Data Lake stream provenance events. Well accepted provenance standards like W3C PROV [6] and OPM [5] represent provenance as a directed acyclic graph (G = (V, E)). A node (v 2 V) can be an activity, entity or agent while an edge (e = hvi, vji where e 2 E and vi, vj 2 V) represents a relationship between two nodes. In our provenance collection model, a provenance event always represents an edge in the provenance graph. For example, if process p generates the data product d, the provenance event adds a new edge (e = hp, di where p, d 2 V) into the provenance graph to represent the 'generation' relationship between activity p and entity d. In addition to capturing usage and generation, additional details like configuration parameters and environment information (e.g., CPU speed, memory capacity, network bandwidth) can be stored as attributes connected to the transformation. Inside each transformation, there can be number of intermediate tasks which may themselves generate intermediate data products. A MapReduce job for instance has multiple map and reduce tasks. Capturing provenance from such internal tasks at a high enough level to be useful helps in debugging and reproducing transformations. When the output data from one analysis tool is used as the input to another, integration of provenance collected from both transformations can be guaranteed only by a consistent lake unique persistent ID policy. This may require a global policy enforced for all contributing organizations to a Data Lake. This unique ID notion could be based on file URLs and randomly generated data identifiers which are appended to data records when producing outputs so that the following transformations can use the same identifiers. It could also be achieved using globally persistent IDs such as the Handle system or DOIs.

D. Reference Architecture

The reference architecture, uses a central provenance collection subsystem. Provenance events captured from components in the Data Lake are streamed into the provenance subsystem where they are processed, stored and analysed. The Provenance Stream Processing and Storage component at the heart of this architecture accepts the stream of provenance notifications (Ingest API) and supports queries (Query API). A live stream processing subsystem supports live queries while storage subsystem persists provenance for long term usage. When long running experiments in the Data Lake produce large volumes of provenance data, stream processing techniques become extremely useful as storing full provenance is not feasible. The Messaging System guarantees reliable provenance event delivery into the central provenance processing layer. Usage subsystem shows how provenance collected around the Data Lake can be used for different purposes. Both live and post-execution queries over collected provenance with Monitoring and Visualization helps in scenarios like the two use cases that we discussed above. There are other advantages as well such as Debugging and Reproducing experiments in the Data Lake. In order to capture information about the origins of data, provenance must be captured at the Ingest. Some data products may carry their previous provenance information which should be integrated as well. Researchers may export data products from the Data Lake in some situations. Such data products should be coupled with their provenance for better usage.

III. PROTOTYPE IMPLEMENTATION

We set up a prototype Data Lake and implemented a use case on top of it to evaluate the feasibility of our reference architecture. We used our provenance collection tools to capture, store, query and visualize provenance in our Data Lake. The reference architecture introduces both stored provenance processing and real time provenance processing for Data Lakes. In this prototype, we implement stored provenance processing; real time provenance processing is future work. The central provenance subsystem uses our Komadu provenance collection framework.

A. Komadu

Komadu is a W3C PROV based provenance collection framework which accepts provenance from distributed components through RabbitMQ4 messaging and web services channels. It does not depend on any global knowledge about the system in a distributed setting. This makes it a good match for a Data Lake environment where different systems are used to perform different data transformations. Komadu API can be used to capture provenance events from individual components of the Data Lake. Each ingest operation adds a new relationship (R) between two nodes (a node can be an activity (A), entity(E) or agent(G)) of the provenance graph being generated. For example, when an activity A generates an entity E, the add Activity Entity Relationship(A, E, R) operation can be used to add a was Generated By relationship between A and E. Using the query operations, full provenance graphs including all connected edges can be generated for Entities, Activities and Agents by passing the relevant identifier. Backward only and forward only provenance graphs can be generated for Entities. In addition to that, Komadu API consists of operations to access the attributes of all types of nodes. Komadu Cytoscape5 plugin can be used to visualize and navigate through provenance graphs.

B. Data Lake Use Case

The Data Lake prototype was implemented using an HDFS cluster and the transformations were performed using Hadoop and Spark. Analysing data from social media to identify trends is commonly seen in Data Lakes. As shown in

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

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

Figure 4, we have implemented a chain of transformations based on Twitter data to first count hash tags and then to get aggregated counts based on categories. Apache Flume6 was used to collect Twitter data and store in HDFS through the Twitter public streaming API. For each tweet, Flume captures the Twitter handle of the author, time, language and the full message and writes a record into an HDFS file. After collecting Twitter data over a period of five days, a Hadoop job was used to count hash tags in the full Twitter dataset. A new HDFS file with hash tag counts is generated as the result of the first Hadoop job which is used by a separate Spark job to get aggregated counts according to categories (sports, movies, politics etc). We just used a fixed set of categories for this prototype implementation to make it simple. In real Data Lakes, these transformations can be performed by different scientists at different times. They may use frameworks based on their preference and expertise. That is why we used two different frameworks for the transformations in our prototype to show how provenance can be integrated across systems.

C. Provenance Queries and Visualization

After executing the provenance enabled Hadoop and Sparkjobs on collected Twitter data, Komadu query API was used to generate provenance graphs. Komadu generates PROVXML provenance graphs and it comes with a Cytoscape plugin which can be used to visualize and explore them. Fine-grained provenance includes input and output datasets for each transformation, intermediate function executions and all intermediate data products generated during the execution. Provenance from Flume, Hadoop and Spark has been integrated together through the usage of unique data identifiers.

D. Performance Evaluation

To build our prototype, we used five small vm instances with 2 cpu cores of 2.5ghz speed, 4 gb of ram and 50 gb local storage on each instance. Four instances were used for the hdfs cluster including one master node and three slave nodes. Total of 3.23 gb twitter data was collected over a period of five days by running flume on the master node. Hadoop and spark clusters were set up on top of our four node hdfs cluster. One separate instance was allocated to set up the provenance subsystem using rabbitmq and komadu tools.

In order to minimize the provenance capture overhead, we used a dedicated thread pool and a provenance event batching mechanism in our client library. When the batch size is set to a relatively large number (>500), execution time becomes almost independent of the thread pool size as the number of messages sent through the network reduces. Therefore, we set the client thread pool size to 5 in each of our experiments. Figure 6a shows how the provenance enabled hadoop execution time for a particular job varies when the batch size is increased from 100 to 30000 (provenance events). As per this result, we set the batch size to 5000 in each of our experiments. We used json format to encode provenance events and the average event size is around 120 bytes. The average size of a batched message sent over the network is around 600 kb (5000 x 120 bytes).

IV. RELATED WORK

Apache falcon7 manages the data lifecycle in hadoop big data stack. Falcon supports creating lineage enabled data processing pipelines by connecting hadoop based processing systems. Apache nifi8 is another data flow tool which captures lineage while moving data among systems. Neither tool captures detailed provenance within transformation steps. Few recent studies target provenance in individual big data processing frameworks like hadoop and spark. Wang j. Et al. Present a way of capturing provenance in mapreduce workflows by integrating hadoop into kepler. Ramp and hadoop prov are two attempts to capture provenance by extending hadoop. Provenance in pache spark and provenance in streaming data have also been studied. Capturing provenance in traditional scientific workflows is another area which has been studied in depth. None of these studies focus on integrating provenance from different frameworks in a shared environment. While any of these big data processing frameworks can be connected to a data lake, as we argued above, a data lake can not depend on such framework specific provenance collection mechanisms due to provenance integration challenges. Therefore, provenance stitching techniques are hard to apply. Wang, j. Et al. Identify the challenges in big data provenance which are mostly applicable in a data lake environment. Distributed big data provenance integration has been identified as a challenge in their work where we present a solution in the context of a data lake. But our reference architecture highlights the power of real time or live provenance processing in data lakes which is also left as a future work.

V. CONCLUSION

The reference architecture for integrated provenance demonstrates early value of data provenance as a lightweight approach to traceability. Future work addresses the viability of the approach in obtaining necessary information without excessive instrumentation or manual intervention. Scalability of the technique is to be further assessed within a real Data Lake environment. Persistent ID solutions have tradeoffs; the suitability of one over another in the Data Lake setting is an open question. We have implemented only the stored provenance processing techniques in the presented prototype.

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

REFERENCES

1. Namdeo, Jyoti, and NaveenkumarJayakumar. "Predicting Students Performance Using Data Mining Technique with Rough Set Theory Concepts." International Journal 2.2 (2014).
2. Jayakumar, D.T. and Naveenkumar, R., 2012. SDjoshi,". International Journal of Advanced Research in Computer Science and Software Engineering," Int. J, 2(9), pp.62-70.
3. Raval, K.S., Suryawanshi, R.S., Naveenkumar, J. and Thakore, D.M., 2011. The Anatomy of a Small-Scale Document Search Engine Tool: Incorporating a new Ranking Algorithm. International Journal of Engineering Science and Technology, 3(7).
4. Naveenkumar, J., Makwana, R., Joshi, S.D. and Thakore, D.M., 2015. Performance Impact Analysis of Application Implemented on Active Storage Framework. International Journal, 5(2).
5. Naveenkumar, J., Keyword Extraction through Applying Rules of Association and Threshold Values. International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), ISSN, pp.2278-1021.
6. Jayakumar, M.N., Zaeimfar, M.F., Joshi, M.M. and Joshi, S.D., 2014. INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING & TECHNOLOGY (IJCET). Journal Impact Factor, 5(1), pp.46-51.
7. Kakamanshadi, G., Naveenkumar, J. and Patil, S.H., 2011. A Method to Find Shortest Reliable Path by Hardware Testing and Software Implementation. International Journal of Engineering Science and Technology (IJEST), ISSN, pp.0975-5462.
8. Archana, R.C., Naveenkumar, J. and Patil, S.H., 2011. Iris Image Pre-Processing And Minutiae Points Extraction. International Journal of Computer Science and Information Security, 9(6), p.171.
9. Salunkhe, R. and Jaykumar, N., 2016, June. Query Bound Application Offloading:Approach Towards Increase Performance of Big Data Computing. In Journal of Emerging Technologies and Innovative Research (Vol. 3, No. 6 (June-2016)). JETIR.
10. Salunkhe, R., Kadam, A.D., Jayakumar, N. and Thakore, D., 2016, March. In search of a scalable file system state-of-the-art file systems review and map view of new Scalable File system. In Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on (pp. 364-371). IEEE.
11. Naveenkumar, J., Makwana, R., Joshi, S.D. and Thakore, D.M., 2015. Offloading Compression and Decompression Logic Closer to Video Files Using Remote Procedure Call.Journal Impact Factor, 6(3), pp.37-45.
12. Jayakumar, N., Singh, S., Patil, S.H. and Joshi, S.D., 2015. Evaluation Parameters of Infrastructure Resources Required for Integrating Parallel Computing Algorithm and Distributed File System. IJSTE-Int. J. Sci. Technol. Eng, 1(12), pp.251-254.
13. Kumar, N., Angral, S. and Sharma, R., 2014. Integrating Intrusion Detection System with Network Monitoring. International Journal of Scientific and Research Publications, 4, pp.1-4.
14. Jayakumar, N., Bhardwaj, T., Pant, K., Joshi, S.D. and Patil, S.H., 2015. A Holistic Approach for Performance Analysis of Embedded Storage Array. Int. J. Sci. Technol. Eng, 1(12), pp.247-250.
15. Jayakumar, N., 2014. Reducts and Discretization Concepts, tools for Predicting Student's Performance. Int. J. Eng. Sci. Innov. Technol, 3(2), pp.7-15.
16. Salunkhe, R., Kadam, A.D., Jayakumar, N. and Joshi, S., 2016, March. Luster a scalable architecture file system: A research implementation on active storage array framework with Luster file system. In Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on (pp. 1073-1081). IEEE.
17. Naveenkumar, J., SDJ, 2015. Evaluation of Active Storage System Realized Through Hadoop. International Journal of Computer Science and Mobile Computing, 4(12), pp.67-73.
18. Bhore, P.R., Joshi, S.D. and Jayakumar, N., 2016. A Survey on the Anomalies in System Design: A Novel Approach. International Journal of Control Theory and Applications, 9(44), pp.443-455.
19. Bhore, P.R., Joshi, S.D. and Jayakumar, N., 2017. Handling Anomalies in the System Design: A Unique Methodology and Solution. International Journal of Computer Science Trends and Technology, 5(2), pp.409-413.
20. Zaeimfar, S.N.J.F., 2014. Workload Characteristics Impacts on file System Benchmarking. Int. J. Adv, pp.39-44.
21. Bhore, P.R., Joshi, S.D. and Jayakumar, N., 2017. A Stochastic Software Development Process Improvement Model To Identify And Resolve The Anomalies In System Design. Institute of Integrative Omics and Applied Biotechnology Journal, 8(2), pp.154-161.
22. Kumar, N., Kumar, J., Salunkhe, R.B. and Kadam, A.D., 2016, March. A Scalable Record Retrieval Methodology Using Relational Keyword Search System. In Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies (p. 32). ACM.
23. Naveenkumar, J. and Joshi, S.D., 2015. Evaluation of Active Storage System Realized Through Hadoop. Int. J. Comput. Sci. Mob. Comput, 4(12), pp.67-73.
24. Naveenkumar, J., Bhor, M.P. and Joshi, S., 2011. A self process improvement for achieving high software quality. International Journal of Engineering Science and Technology (IJEST), 3(5), pp.3850-3053.
25. Naveenkumar, J. and Raval, K.S., 2011. Clouds Explained Using Use-Case Scenarios.INDIACom-2011 Computing for Nation Development, 3.
26. Sawant, Y., Jayakumar, N. and Pawar, S.S., 2016. Scalable Telemonitoring Model in Cloud for Health Care Analysis. In International Conference on Advanced Material Technologies (ICAMT) (Vol. 2016, No. 27th).
27. Naveenkumar, J. and Joshi, S.D., 2015. Evaluation of Active Storage System Realized through MobilityRPC.
28. kumarSingha, A., Patilb, S.H. and Jayakumarc, N., A Survey of Increasing I/O Latency in I/O Stack.
29. Bhore, P.R., Joshi, S.D. and Jayakumar, N., 2016. A Survey on the Anomalies in System Design: A Novel Approach. International Journal of Control Theory and Applications, 9(44), pp.443-455.
30. Singh, A.K., Pati, S.H. and Jayakumar, N., A Treatment for I/O Latency in I/O Stack.
31. Jaiswal, U., Pandey, R., Rana, R., Thakore, D.M. and JayaKumar, N., Direct Assessment Automator for Outcome Based System.
32. Kulkarnia, A. and Jayakumarb, N., A Survey on IN-SITU Metadata Processing in Big Data Environment.
33. Jayakumar, N., Iyer, M.S., Joshi, S.D. and Patil, S.H., A Mathematical Model in Support of Efficient offloading for Active Storage Architectures.