Provenance as Essential Infrastructure for Data Lakes [Preprint, forthcoming in IPAW 2016]

Isuru Suriarachchi and Beth Plale

School of Informatics and Computing, Indiana University, Bloomington, IN, USA
{isuriara,plale}@cs.indiana.edu

Abstract. The Data Lake is emerging as a Big Data storage and management solution which can store any type of data at scale and execute data transformations for analysis. Higher flexibility in storage increases the risk of Data Lakes becoming data swamps. In this paper we show how provenance contributes to data management within a Data Lake infrastructure. We study provenance integration challenges and propose a reference architecture for provenance usage in a Data Lake. Finally we discuss the applicability of our tools in the proposed architecture.

1 Introduction

Big Data has brought about recognition in industry and research alike that data can be profitably mined for insight and forecasts. Data from numerous sources (e.g., clickstream, sensor data, social media, server logs) are being brought together. The Data Lake [5] has been introduced as an infrastructure which supports broader analysis on various types of data from different sources. It can store unstructured, semi-structured, and structured data at scale and support data transformations by integrating Big Data processing frameworks such as Apache Hadoop[1] and Apache Spark[2]. As the Data Lake does not enforce a schema at the time of ingest, scientists can easily dump data from various sources and process them only when necessary. This "dump everything" nature in a Data Lake increases the flexibility of data storage. However without some level of organization, the popular literature goes, the Data Lake will turn into a data swamp [2]. Transformations performed on data products in a Data Lake write their results back into the lake. A data product can go through number of transformations during its lifecycle within a Data Lake. Critical focus of our attention is on using provenance and lineage information in Data Lakes to avoid data swamps.

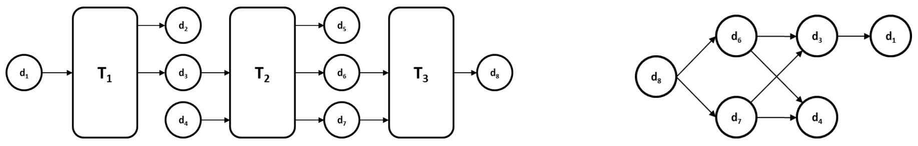
We propose two use cases to motivate the study of provenance in a Data Lake. *Use Case 1*: Suppose sensitive data are ingested by a scientist into a Data Lake. By definition of the data lake, the sensitive data will likely undergo schema translation before being used by someone else. Can provenance be used to determine whether the schema and schema translation process change the sensitivity level of the data? Can this be determined quickly enough to take

[1] http://hadoop.apache.org/
[2] http://spark.apache.org/

appropriate action, and if so, what actions should be taken? *Use Case 2*: Using provenance to assess-respond in real time: Repeating a Big Data transformation in a Data Lake is expensive due to high resource and time consumption. Can we use live streaming provenance from experiments to monitor them real time and identify the faults early in the execution?

2 Provenance Capture in a Data Lake

If a Data Lake could somehow ensure that every data product in the lake is connected with its provenance starting from the origin, critical traceability can be achieved. This is challenging because a data product may go through different distributed processing systems during its lifecycle. Processing frameworks used around a Data Lake can include batch processing systems, stream processing systems, traditional workflow engines or even legacy scripts. These frameworks may or may not produce provenance by default. Even if there are provenance collection techniques [1] for certain systems, they may use their own ways of storing provenance or use different standards. Therefore generating integrated provenance traces is tough. Stitching techniques [3] exist which bring all provenance traces into a common model and then integrate them together. However there are certain limitations in such techniques like loss of information during conversions and higher computation overheads for large provenance graphs which are common in Data Lakes. In addition to that, real time provenance integration (use case 2) can not be achieved by such post processing techniques. As a solution for this provenance integration problem, we propose a central provenance collection system to which all distributed components within the Data Lake stream provenance events. For each transformation, the data scientist who writes the data processing code can instrument her code to generate provenance at all needed steps.



(a) Data Flow Across Transformations (b) Data Lineage

Fig. 1. Provenance for Series of Transformations

Provenance is commonly represented as a directed acyclic graph ($G = (V, E)$). A node ($v \in V$) can be an activity, entity or agent while an edge ($e = \langle v_i, v_j \rangle$ where $e \in E$ and v_i, $v_j \in V$) 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 ($e = \langle p, d \rangle$ where p, $d \in V$) into the provenance graph to represent the 'generation' relationship between activity p and entity d. When all systems connected to the Data Lake continue to send provenance events, the central provenance collection system keeps adding new edges into the provenance graph. Provenance integration across distributed components is guaranteed by using unique identifiers for all data products within the Data Lake. As a simple example, consider the data flow diagram in Figure 1a. The data product d_1 is subjected to transformation T_1 and it generates data products d_2 and d_3 as results. T_2 uses d_3 together with a new data product d_4 and generates d_5, d_6 and d_7. Finally T_3 uses d_6 and d_7 and generates d_8 as the final output. When all three transformations T_1, T_2 and T_3 have sent provenance events, complete provenance graph is created in the central provenance collection system. Figure 1b shows the provenance graph which represents the data flow when queried for final output d_8. Details like scientists involved, configuration parameters and environment information (CPU speed, memory capacity, network bandwidth etc.) can also be captured as provenance.

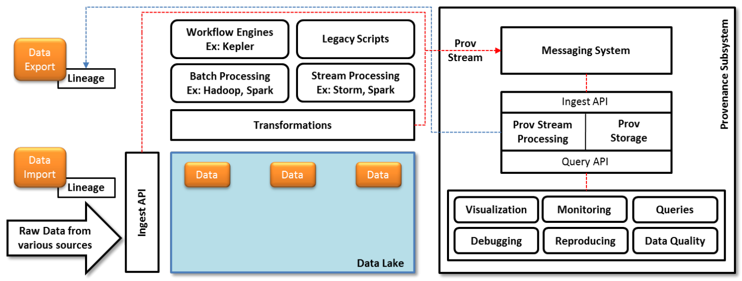


Fig. 2. Provenance for Data Lakes: Reference Architecture

Figure 2 shows the reference architecture that we propose for Data Lakes based on the provenance integration technique discussed above. Provenance Stream Processing and Storage component is the heart of this architecture which accepts the stream of provenance notifications through its Ingest API and supports queries through its Query API. Live stream processing sub-system supports live queries while storage sub-system persists provenance for long term usage. The Messaging System guarantees reliable provenance event delivery into the central provenance storage. Various distributed transformation tools around the Data Lake stream provenance events into the central Provenance Subsystem. Transformation logics have to be instrumented to capture provenance at required granularity. In order to capture information about the origins of the data products, provenance must be captured at the Ingest. Some data products may carry their previous provenance information which should be integrated as well. Scientists may export data products from the Data Lake in some situations. Such data products should be coupled with their provenance for better usage. 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.

Komadu [4] is a W3C PROV based provenance collection framework whose design and API are not coupled to any specific system and can be used as a general provenance collection framework. Capturing provenance from distributed applications is made easy with Komadu as it does not depend on any global knowledge about the system. This makes it applicable in the above architecture to capture provenance in a Data Lake. Komadu provides connectors to plug its Ingest API with the RabbitMQ[3] messaging system. The Komadu toolkit includes efficient client libraries for java and javascript applications which minimize instrumentation overhead. Provenance storage system in Komadu is designed based on a relational data model and implemented using MySQL. Ingested provenance events are asynchronously processed and stored in relational tables. Graph generation is delayed till query time to reduce computation at ingest time. Komadu toolkit comes with a Cytoscape[4] plugin as well which can visualize generated provenance graphs.

3 Final Remarks and Future Work

Although Komadu seems to fit well in our reference architecture for provenance capture in a Data Lake, it supports only queries over stored provenance and lacks live provenance stream processing. Fine-grained provenance captured from massively parallel systems can produce large amounts of provenance data that leads to the "Big Provenance" problem [6]. As future work, we plan to combine a series of Big Data transformations to replicate a Data Lake environment and apply our reference architecture into the system using Komadu to see how it performs. We focus on solving the Big Provenance problem using real time provenance stream processing algorithms which reduce the amount of stored provenance.

4 Acknowledgement

This work is funded in part by a grant from the NSF, ACI-0940824.

References

1. Akoush, S., Sohan, R., Hopper, A.: Hadoopprov: Towards provenance as a first class citizen in mapreduce. In: TaPP. pp. 11:1–11:4 (2013)
2. Chessell, M., Scheepers, F., Nguyen, N., van Kessel, R., van der Starre, R.: Governing and managing big data for analytics and decision makers (2014), http://www.redbooks.ibm.com/redpapers/pdfs/redp5120.pdf
3. Missier, P., Ludascher, B., Bowers, S., Dey, S., Sarkar, A., Shrestha, B., Altintas, I., Anand, M., Goble, C.: Linking multiple workflow provenance traces for interoperable collaborative science. In: WORKS. pp. 1–8 (Nov 2010)
4. Suriarachchi, I., Zhou, Q., Plale, B.: Komadu: A capture and visualization system for scientific data provenance. Journal of Open Research Software 3(1) (2015)
5. Terrizzano, I., Schwarz, P.M., Roth, M., Colino, J.E.: Data wrangling: The challenging journey from the wild to the lake. In: CIDR (2015)
6. Wang, J., Crawl, D., Purawat, S., Nguyen, M., Altintas, I.: Big data provenance: Challenges, state of the art and opportunities. In: Big Data. pp. 2509–2516 (2015)

[3] http://www.rabbitmq.com/
[4] http://www.cytoscape.org/