Crossing Analytics Systems: A Case for Integrated Provenance in Data Lakes

12th eScience 2016 IEEE INTERNATIONAL CONFERENCE

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

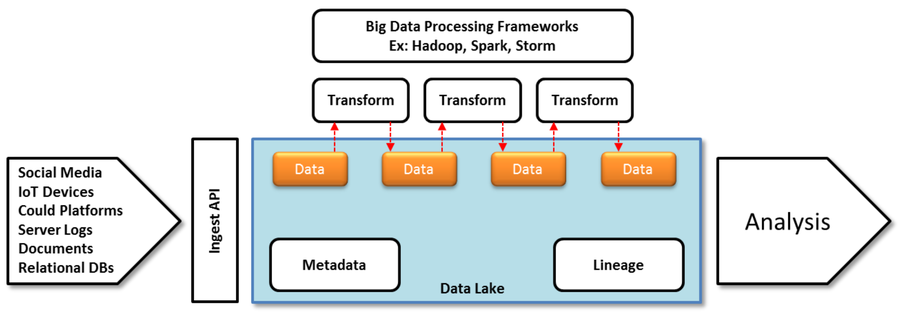
Isuru Suriarachchi and Beth Plale

School of Informatics and Computing, Indiana University

MOTIVATION

The Data Lake

- Gathers data from various sources like sensor data, social media, cloud platforms and server logs
- Supports structured, semi-structured or unstructured data with no schema enforced at ingest time
- Data products are subjected to series of data transformations through distributed Big Data processing frameworks like Apache Hadoop and Apache Spark for analysis



Problem

- Increased flexibility leads to harder manageability
- Can provenance contribute to safer and more efficient Data Lakes through real time assess-respond and post execution traceability?

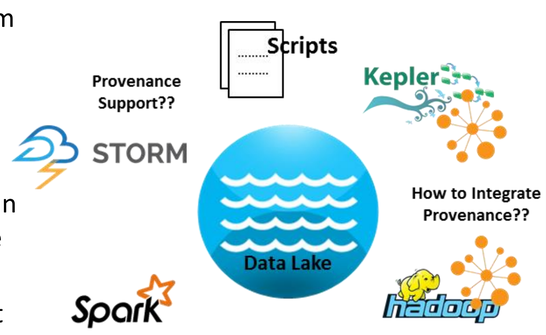
Provenance Use Cases in Data Lake

- **Use Case 1:** Suppose sensitive data are deposited into a Data Lake; social science survey data for instance. This dataset is processed by a chain of transformations. Can data provenance prevent improper leakage of sensitive parts into derived data?
- **Use Case 2:** Repeating a Big Data transformation in a Data Lake is expensive due to high resource and time consumption. Can live streaming provenance from experiments identify problems early in their execution?

PROVENANCE INTEGRATION

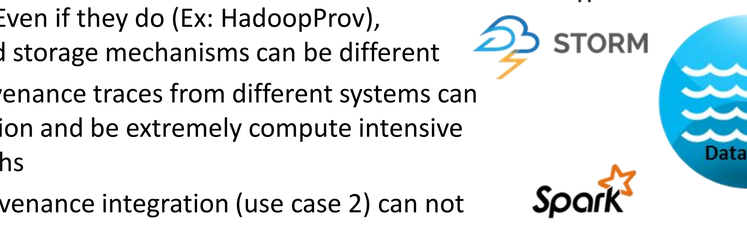
Challenge

- Comprehensive provenance should be integrated from ingest through all distributed transformations
- Transformation systems may or may not produce provenance. Even if they do (Ex: HadoopProv), standards and storage mechanisms can be different
- Stitching provenance traces from different systems can lose information and be extremely compute intensive for large graphs
- Real time provenance integration (use case 2) can not be achieved by post processing techniques



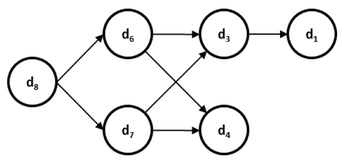
Methodology

- Central provenance collection sub system to which all distributed components within the Data Lake stream provenance events
- Provenance integration across distributed components is guaranteed by using unique identifiers for all data products within the Data Lake
- 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 $(e = < v_i, v_j > where\ e \in E\ and v_i, v_j \in V)$ is a relationship between two nodes
- In our model, a provenance ***event*** always represents an ***edge*** in the provenance graph
- Ex: If process p generates data product d, the provenance event adds a new edge $(e = < p, d > where\ p, d \in V)$ into the provenance graph representing 'generation' relationship

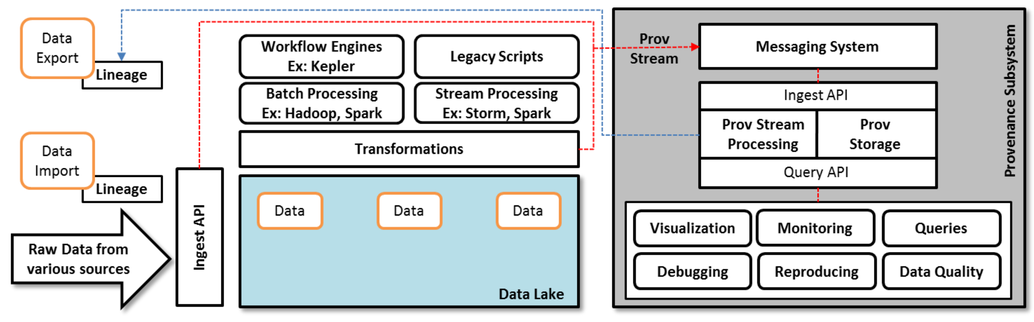


A Simplified Data Flow Scenario in a Data Lake

- Data product d_1 is subjected to transformation T_1 and it generates d_2 and d_3. T_2 uses d_3 with a new data product d_4 to generate d_5, d_6 and d_7. Finally T_3 uses d_6 and d_7 to generate d_8
- Individual distributed transformations T_1, T_2 and T_3 stream provenance notifications to the central provenance store
- Figure shows the provenance graph which represents the data lineage of the data product

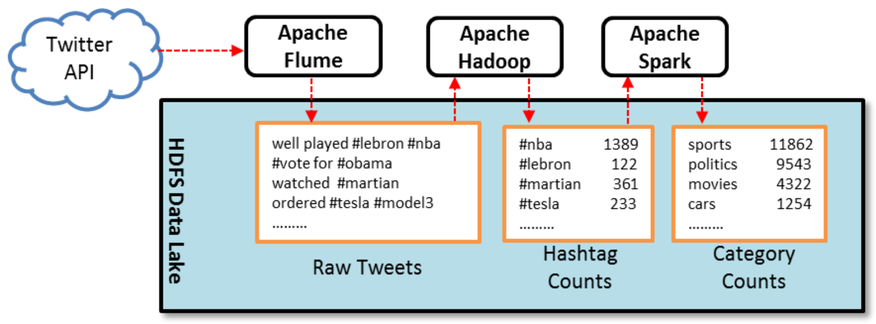


REFERENCE ARCHITECTURE



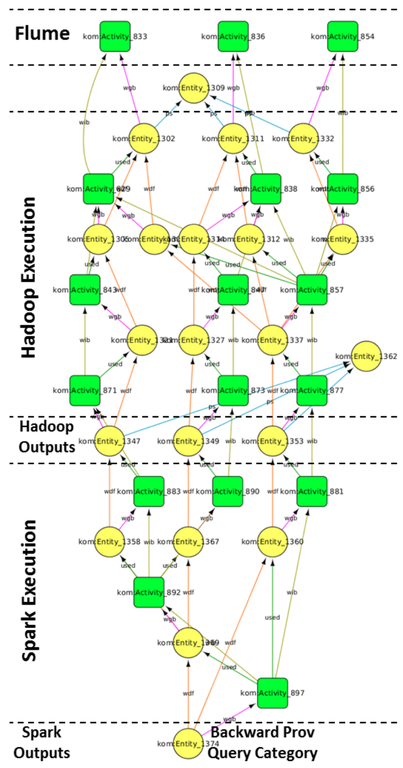
- Provenance Stream Processing and Storage sub-system: heart of system; accepts provenance notifications through Ingest API and supports queries through Query API
- Messaging System: guarantees reliable message delivery into Provenance Storage
- Instrumented transformations: stream provenance events into Provenance Subsystem
- To capture information about origins of data products, provenance must be captured at Ingest
- Exported data products should be coupled with their provenance for better usability
- 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

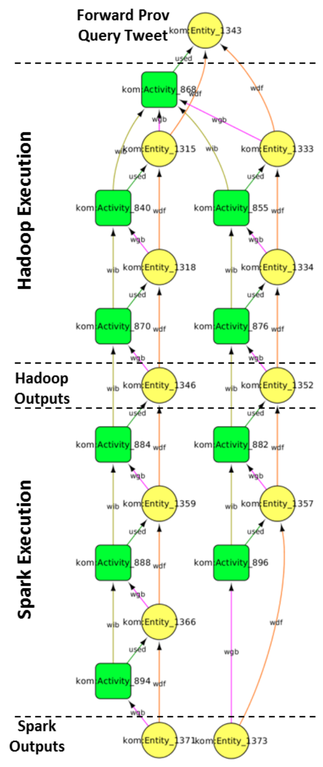
PROTOTYPE IMPLEMENTATION



- **Use Case:** A Twitter data processing chain
- Three different frameworks used for data processing
- Komadu as the central provenance store
- Flume, Hadoop and Spark jobs were instrumented using Komadu client libraries
- Generated UUIDs are assigned for data products and persisted with them

Integrated Provenance Traces





FUTURE WORK

- Live provenance stream processing for real time monitoring and computational steering
- Efficient provenance graph storage/querying to handle "Big Provenance"
- Better usage of persistent identifiers through a Handle System or DOIs

CONTACT

- Isuru Suriarachchi, isuriara@indiana.edu
- Beth Plale, plale@indiana.edu
- Data to Insight Center, http://d2i.indiana.edu/