Crossing Analytics Systems: Case for Integrated Provenance in Data Lakes

Isuru Suriarachchi and Beth Plale
School of Informatics and Computing and Computing
Indiana University

IEEE E-science 2016 : Hot Topics
The **Data Lake** has arisen within last couple of years as conceptualization of data management framework with flexibility to support multiple data processing tools needed for truly Big Data analytics.
Data Warehouse

• Supports multidimensional analytical processing
  – Online Analytical Processing (OLAP) or Multidimensional OLAP

• Numeric facts (measures) categorized by dimensions creating vector space (OLAP cube).

• Interface is matrix interface like Pivot tables

• Schema is star schema, snowflake schema

• Storage is largely relational database
Data Warehouse Architecture

- ETL: Extraction, Transformation, Load

Credit: https://www.linkedin.com/topic/data-warehouse-architecture
Challenging the Warehouse: Big Data

• From numerous sources
  – social media, sensor data, IoT devices, server logs, clickstream etc.

• Not all numeric (quantitative) thus differently structured
  – Structured, semi-structured, unstructured

• Continuously generated or archived
Suitability of Data Warehouse for Today’s Big Data

• ETL imposes burden
  – Schema on write
  – Inflexibility/inefficiency at ingest time
  – Information loss upon schema translation

• Weak fit for popular Big Data analytical tools (e.g., Spark, Hadoop) and data serving platforms (e.g., HDFS, S3)
Data Lake

- A scalable storage infrastructure with no schema enforcement at ingest
- Data ingested in raw form: no loss
- Schema-on-read
- Integrated Transformations
  – With e.g., Hadoop, Spark
Data Lake Challenges

• Increased flexibility leads to harder manageability
  – Differently typed data can be easily dumped into the Data Lake
  – Data products can be in different stages of their lifecycle: raw, half processed, processed etc.
  – Can easily turn into “data swamps”

• Requires traceability!!..
  – Provenance can help
Data Provenance

• Information about activities, entities and people who involved in producing a data product
• Standards
  – OPM
  – PROV
• If a Data Lake ensures that every data product’s provenance is in place starting from data product’s origin, critical traceability can be had
What provenance perspective could bring to a Data Lake?

- Track origins of data, chained transformations
- Contribute to reuse determinations of trust and quality
- React!! Minimally constrain what enters a Lake?
Challenges in Provenance Capturing

• Chains of Transformations
  – Different analytics systems: Hadoop, Spark etc.

• Need is end to end **integrated provenance across transformations**

• System specific provenance collection methods are less useful
  – Integration/stitching problems
  – E.g.: RAMP, HadoopProv for Hadoop
Solution to minimal lake governance

• All components in lake *stream* provenance to central provenance subsystem
  – Stores provenance for long term queries
  – Monitors provenance stream in real time
• *Event* in stream represented by *edge* in provenance graph
• Global lake wide policy: Uniform Persistent ID (PID) (Handle, UUIDs, DOIs) attached to all data objects in Data Lake
  – required to guarantee integrated provenance
• PID assigned to all data objects
  – granularity
• Transformations $T_1$, $T_2$, and $T_3$
  – Distributed
  – May use different frameworks

Backward provenance from central provenance store
Provenance traces integrate across systems of Data Lake
Reference Architecture

- Real-time provenance stream processing
- Stored provenance for long term usage
Prototype Use Case

- Different frameworks used
  - Flume: Captures tweets and write into HDFS
  - Hadoop Job: Computes hashtag counts
  - Spark Job: Computes category counts
Central provenance store

• Uses Komadu
  – A distributed provenance collection tool
  – Visualization, Custom Queries

• Log4j like API for provenance capture
• Dedicated thread pool in provenance layer
• Batching to minimize network overhead
Use case evaluation

- Flume, Hadoop and Spark jobs instrumented using Komadu client libraries
- Jobs stream provenance events into central provenance store (Komadu)
- Persistent IDs (UUID) assigned for each data object at entry to data lake; PID persists thereafter with data object
Use case evaluation: experimental environment

- 5 small VM instances, 2 2.5Ghz cores, 4 GB RAM, 50 GB local storage
- 4 VM instances used for HDFS cluster
- 3.23 GB Twitter data collected over 5 days running Flume on master node
- Hadoop and Spark set up on top of HDFS cluster
- Separate instance for RabbitMQ and Komadu
Use case evaluation: Metrics

• Batch size:
  – impact of batch size on provenance capture efficiency. Measured by total execution time for Hadoop using provenance event batching mechanism in Komadu library

• Overhead of provenance capture:
  – Measured against total tool-specific execution time
  – measure overhead of customized value field (in key value pair)
  – Measure overhead of provenance capture for Hadoop and Spark
Batch Size Test

- Hadoop job execution times with varying batch sizes
- Optimal batch size: ~5000 KB
• custom val: emits PID with key value pair as (\#nba, <2, id>) instead of (\#nba, 2)
• data prov HDFS: writes provenance into HDFS, used by HadoopProv and RAMP
Overhead: Spark

- Higher provenance capture overhead compared to Hadoop
Future Work

• Performance overhead is prohibitively high
  – decouple PID assignment from execution?
  Examine granularity

• Live provenance stream processing for real time monitoring/reaction

• Explore minimal provenance at on-line rates and more comprehensive provenance at off-line rates
Work funded in part by National Science Foundation OCI-0940824