Influence-Based Provenance for Dataflow Applications with Taint Propagation

Jason Teoh[1], Muhammad Ali Gulzar[2], Miryung Kim[1]
[1]University of California, Los Angeles, [2]Virginia Tech

Problem Statement

What inputs are responsible for producing suspicious output?

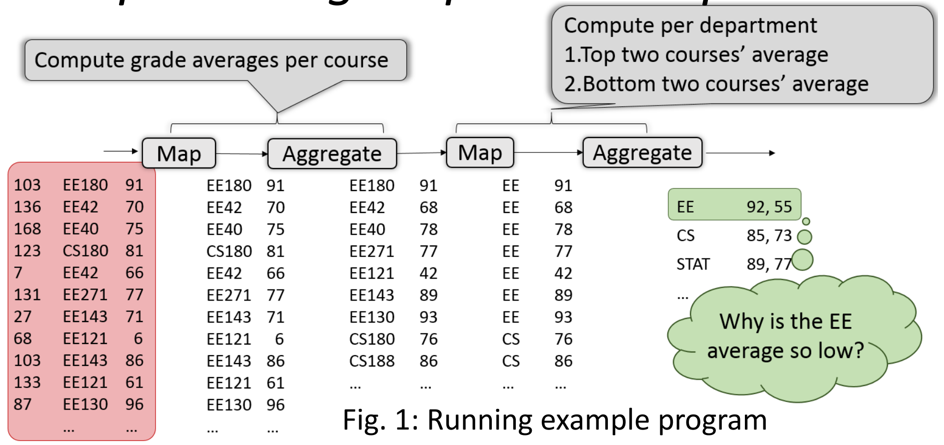
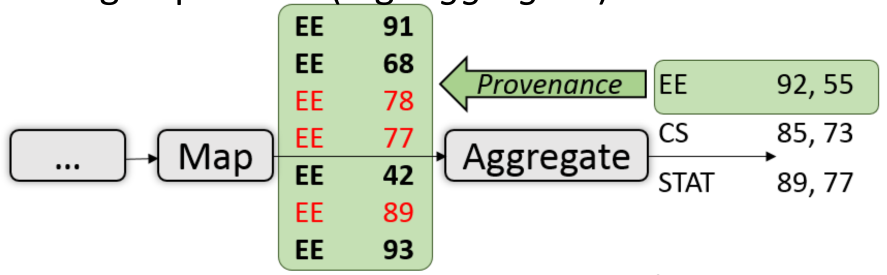

Compute grade averages per course
Compute per department
1.Top two courses' average
2.Bottom two courses' average
Map
Aggregate
Map
Aggregate
EE 92, 55
CS 85, 73
STAT 89, 77
Why is the EE average so low?

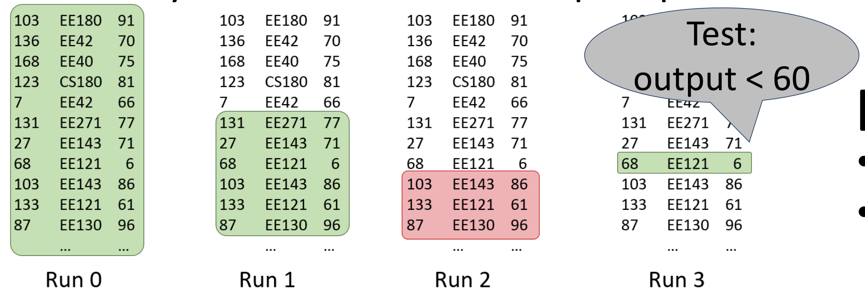
Fig. 1: Running example program

Current State of the Art

- **Data Provenance**: Trace the movement of records through operators (e.g. *Aggregate*)



- **Delta Debugging**: Use an output test function to guide binary search reduction of input space.



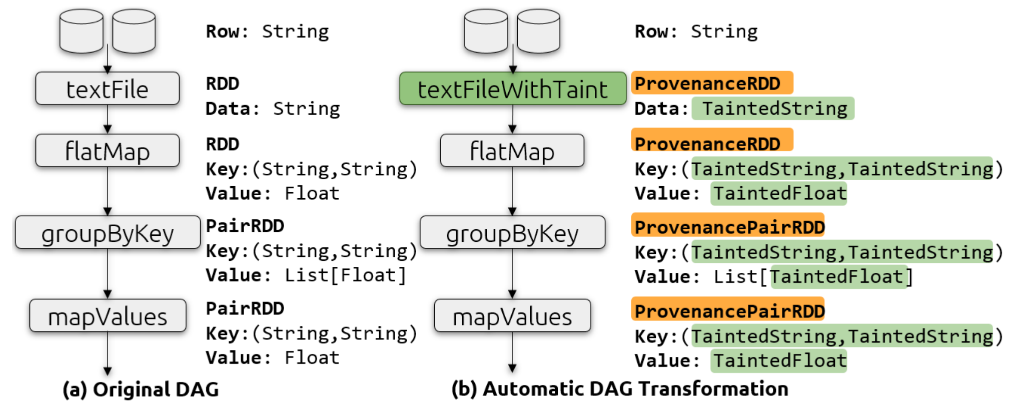
Key Insight

FlowDebug improves provenance **precision** by tracking **input contribution** within **UDFs.**

- **UDF-Aware Tainting** and **Influence Functions** can be used together to improve provenance trace precision.

Novelty 1: UDF Tainting

FlowDebug automatically tracks UDF control and data flow through instrumented data types.

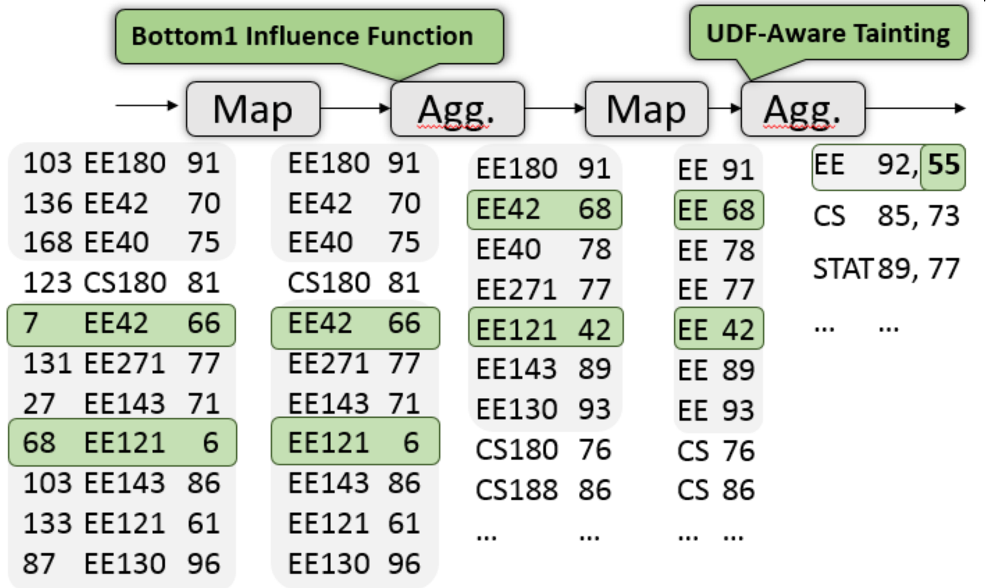


Novelty 2. Influence Function

FlowDebug extends Spark's *combineByKey* API with *Influence Functions* to define flexible, user-defined provenance.

<<interface>> InfluenceFunction

+ init(): InfluenceFunction
+ mergeValue(V): InfluenceFunction
+ mergeFunc(InfluenceFunction): InfluenceFunction
+ finalize(): Provenance



Evaluation Results

Comparisons against Titian (Provenance), BigSift (Delta Debugging), and Spark (baseline)

[RQ1] Precision

- 15-100% precision improvement vs Titian
- 96.8-99.3% recall improvement vs BigSift

[RQ2] Instrumentation Overhead

- 5.4-8X faster with Influence Functions
- 50% overhead with UDF-Aware Tainting
- 0.4-6.1X overhead vs Spark

[RQ3] Tracing Time

- 12-73X, 374-1506X faster than Titian and BigSift
- Tracing at most 25% of total job