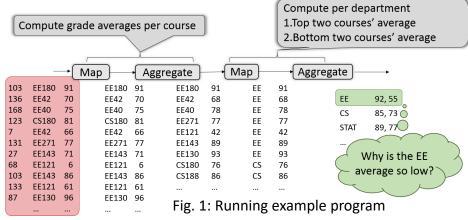
Influence-Based Provenance for Dataflow Applications with Taint Propagation

Jason Teoh¹, Muhammad Ali Gulzar², Miryung Kim¹ ¹University of California, Los Angeles, ²Virginia Tech

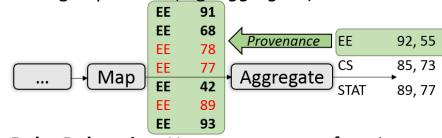
Problem Statement

What inputs are responsible for producing suspicious output?



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.

103 136	EE180 EE42	91 70		103 136	EE180 EE42	91 70	103 136	EE180 EE42	91 70	Test:
168	EE40	75		168	EE40	75	168	EE40	75	
123	CS180	81		123	CS180	81	123	CS180	81	output < 60
7	EE42	66		7	EE42	66	7	EE42	66	7 EE4z
131	EE271	77		131	EE271	77	131	EE271	77	131 EE271
27	EE143	71		27	EE143	71	27	EE143	71	27 EE143 71
68	EE121	6		68	EE121	6	68	EE121	6	68 EE121 6
103	EE143	86		103	EE143	86	103	EE143	86	103 EE143 86
133	EE121	61		133	EE121	61	133	EE121	61	133 EE121 61
87	EE130	96	(87	EE130	96	87	EE130	96	87 EE130 96
)								
I	Run O			Ru	un 1		R	un 2		Run 3

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.

	-			
	Row: String		Row: String	
textFile	RDD	textFileWithTaint	ProvenanceRDD	
	Data: String		Data: TaintedString	
flatMap	RDD	flatMap	ProvenanceRDD	
	Key:(String,String)		<pre>Key:(TaintedString,TaintedString)</pre>	
	Value: Float		Value: TaintedFloat	
groupByKey	PairRDD	groupByKey	ProvenancePairRDD	
groupbykey	<pre>Key:(String,String)</pre>	groupbykey	<pre>Key:(TaintedString,TaintedString)</pre>	
	Value: List[Float]		Value: List[TaintedFloat]	
mapValues	PairRDD	[mapValues]	ProvenancePairRDD	
	<pre>Key:(String,String)</pre>		<pre>Key:(TaintedString,TaintedString)</pre>	
	Value: Float	Ļ	Value: TaintedFloat	
(a) Original D/	AG	(b) Automatic DAG Transformation		
-				

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

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

Bottom1 In	UDF-Aware Tainting		
→ Ma	p → Ag	g. → Ma	p→Agg.→
103 EE180 91	EE180 91	EE180 91	EE 91 EE 92, 55
136 EE42 70	EE42 70	EE42 68	EE 68 CS 85, 73
168 EE40 75	EE40 75	EE40 78	FF 78
123 CS180 81	CS180 81	EE271 77	EE 77 STAT 89, 77
7 EE42 66	EE42 66	EE121 42	EE 42 ··· ···
131 EE271 77	EE271 77	EE143 89	EE 89
27 EE143 71	EE143 71	EE130 93	EE 93
68 EE121 6	EE121 6	CS180 76	CS 76
103 EE143 86	EE143 86	CS188 86	CS 86
133 EE121 61	EE121 61		
87 EE130 96	EE130 96		

Evaluation Results

Comparisons against Titian (Provenance), BigSift (Delta Debugging), and Spark (baseline) [RQ1] Precision [RQ2] Instrumentation Overhead

- •15-100% precision improvement vs Titian •5.4-8X faster with Influence Functions
- •96.8-99.3% recall improvement vs BigSift •50% overhead with UDF-Aware Tainting
 - •0.4-6.1X overhead vs Spark

Novelty 2. Influence Function

<<interface>> InfluenceFunction

[RQ3] Tracing Time •12-73X, 374-1506X faster than Titian and **BigSift** •Tracing at most 25% of total job