Debugging Big Data Analytics in Apache Spark with *BigDebug*

MATTEO INTERLANDI - MUHAMMAD ALI GULZAR

Debugging Cloud Computing Programs

Open Source Data-Intensive Scalable Computing (DISC) Platforms: Hadoop MapReduce and Spark

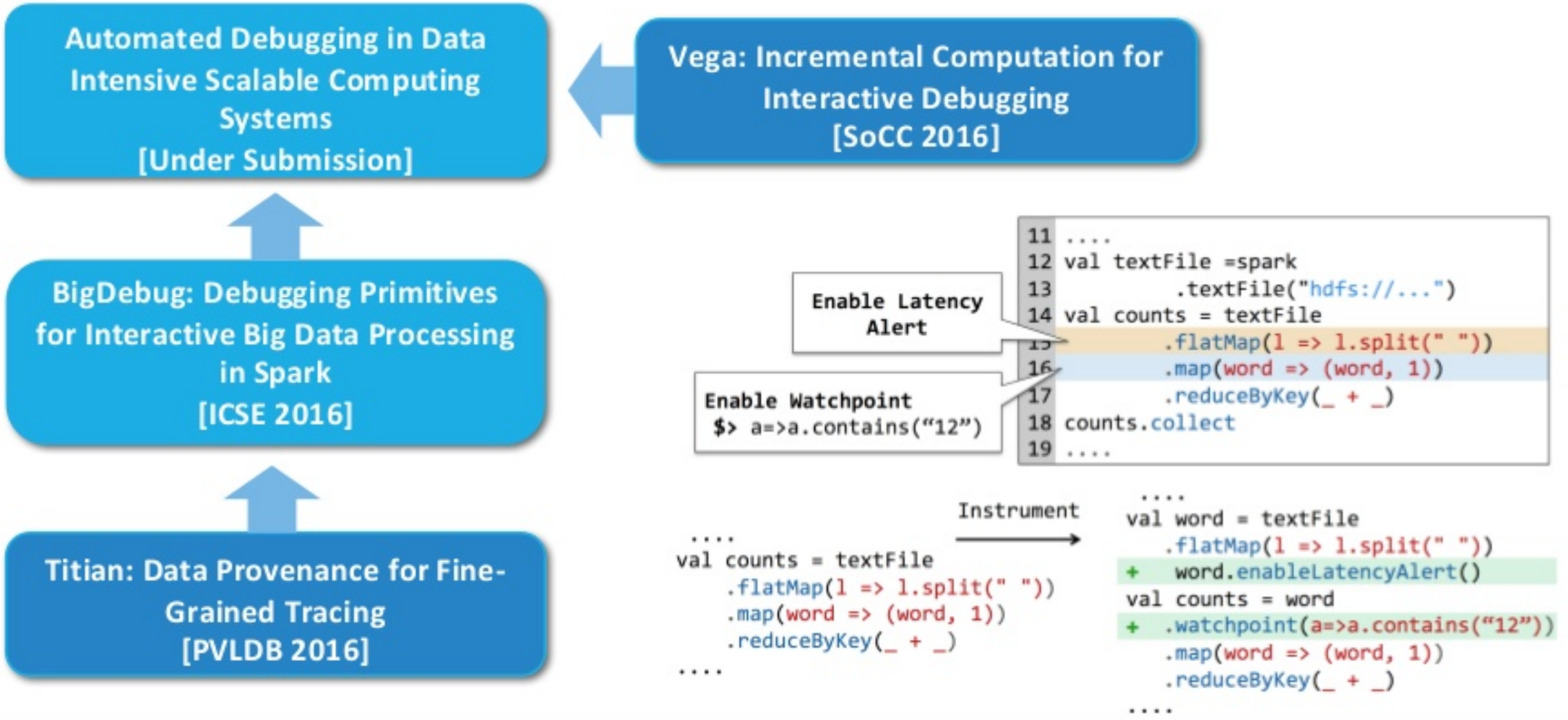
- **Severe lack of debugging** support in these systems
- Programs (i.e., queries, jobs) are batch executed / black boxes

So what to do?

- **Trial and error** debugging on subsample
- **Post-mortem analysis** of error logs
- Analyze **physical view** of the execution (a job id, failed node, etc).

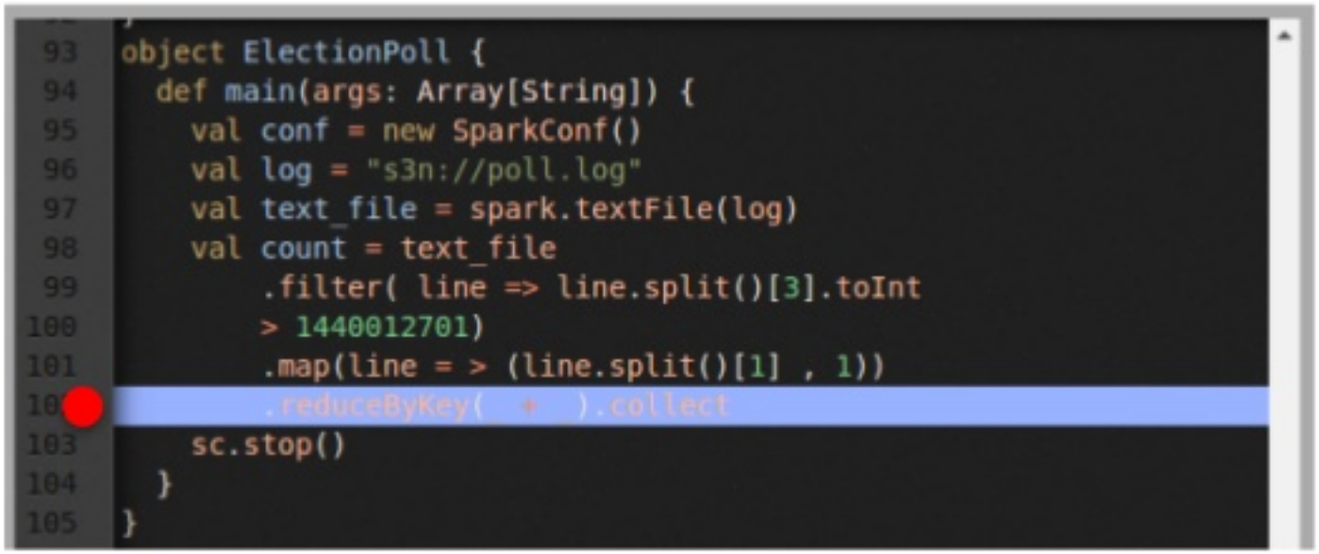
BigDebug Project Overview

Collaboration with Tyson Condie, Miryung Kim, and Todd Millstein



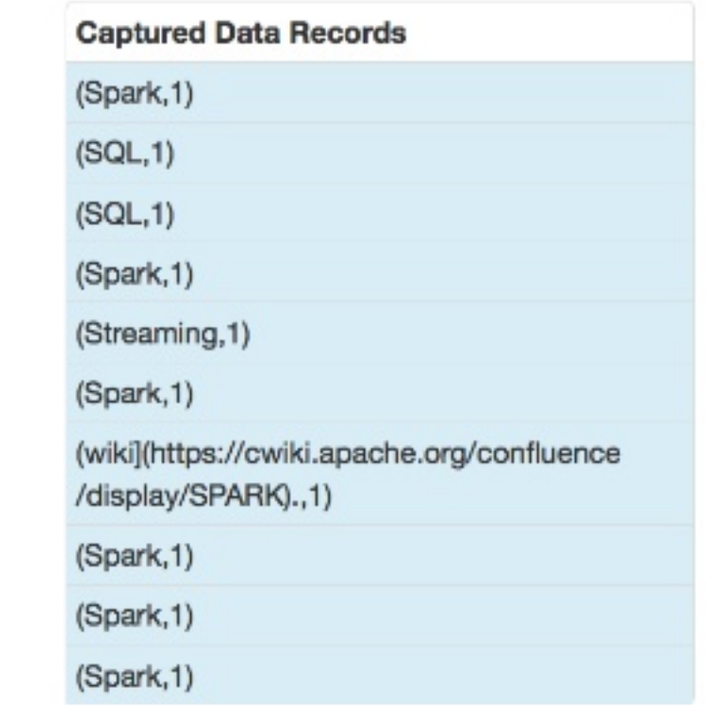
BigDebug: Interactive Debugger Features

1. Simulated Breakpoint



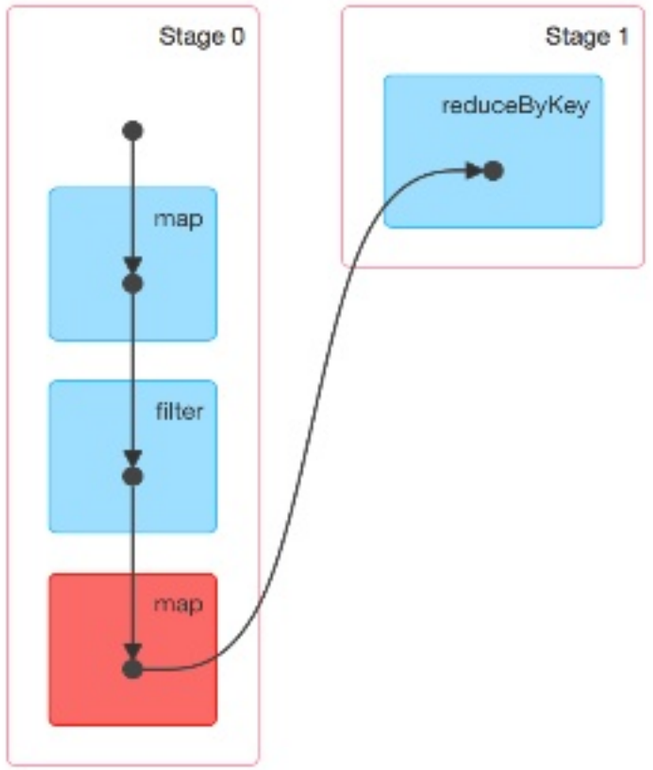
BigDebug: Interactive Debugger Features

1. Simulated Breakpoint
2. Guarded Watchpoint



BigDebug: Interactive Debugger Features

1. Simulated Breakpoint
2. Guarded Watchpoint
3. Crash Culprit Identification



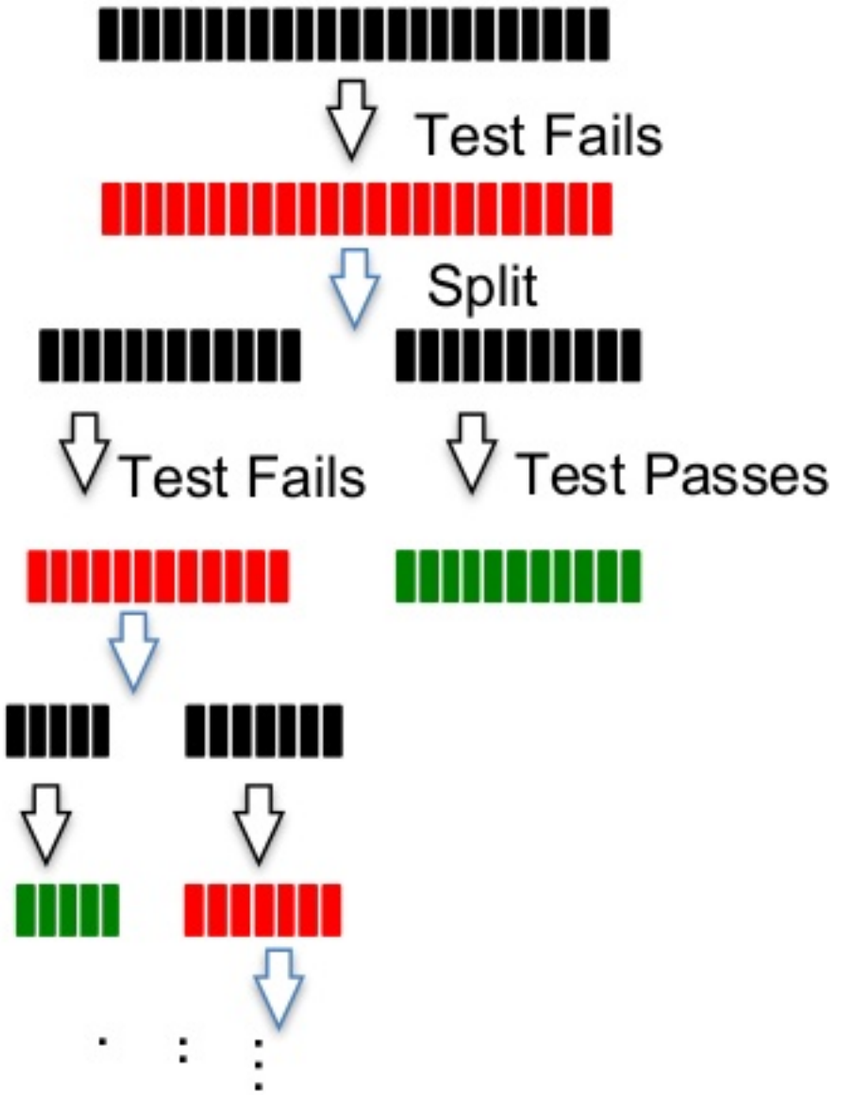
BigDebug: Interactive Debugger Features

1. Simulated Breakpoint
2. Guarded Watchpoint
3. Crash Culprit Identification
4. Backward Tracing

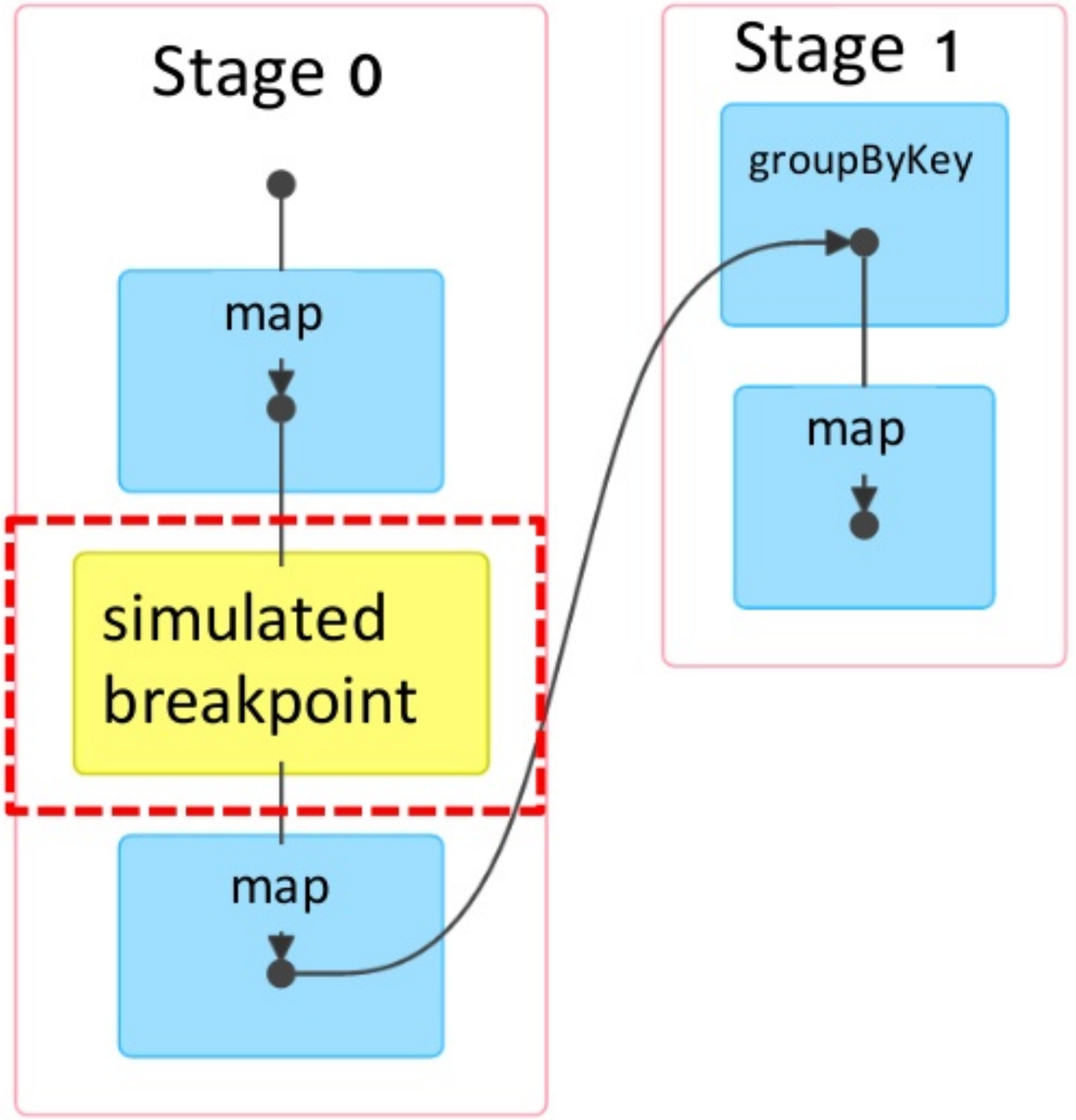
```
$>Crash inducing input records:

9K23 Cruz TX 1440023645
2FSD Cruz KS 1440026456
9909 Cruz KS 1440023768
```

BigDebug: Interactive Debugger Features

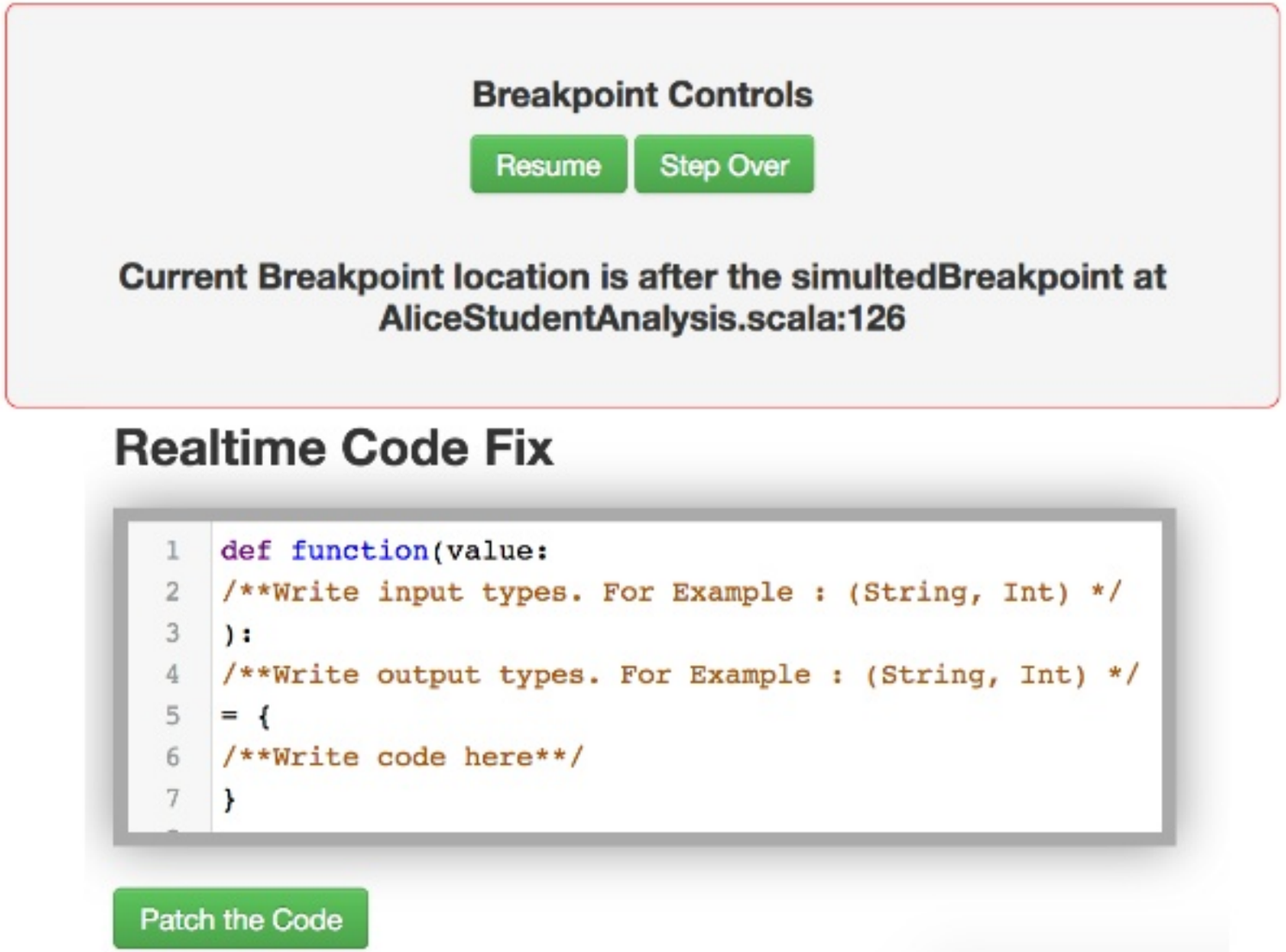
1. Simulated Breakpoint
2. Guarded Watchpoint
3. Crash Culprit Identification
4. Backward Tracing
5. Automated Fault Localization



Feature 1:
Simulated Breakpoint

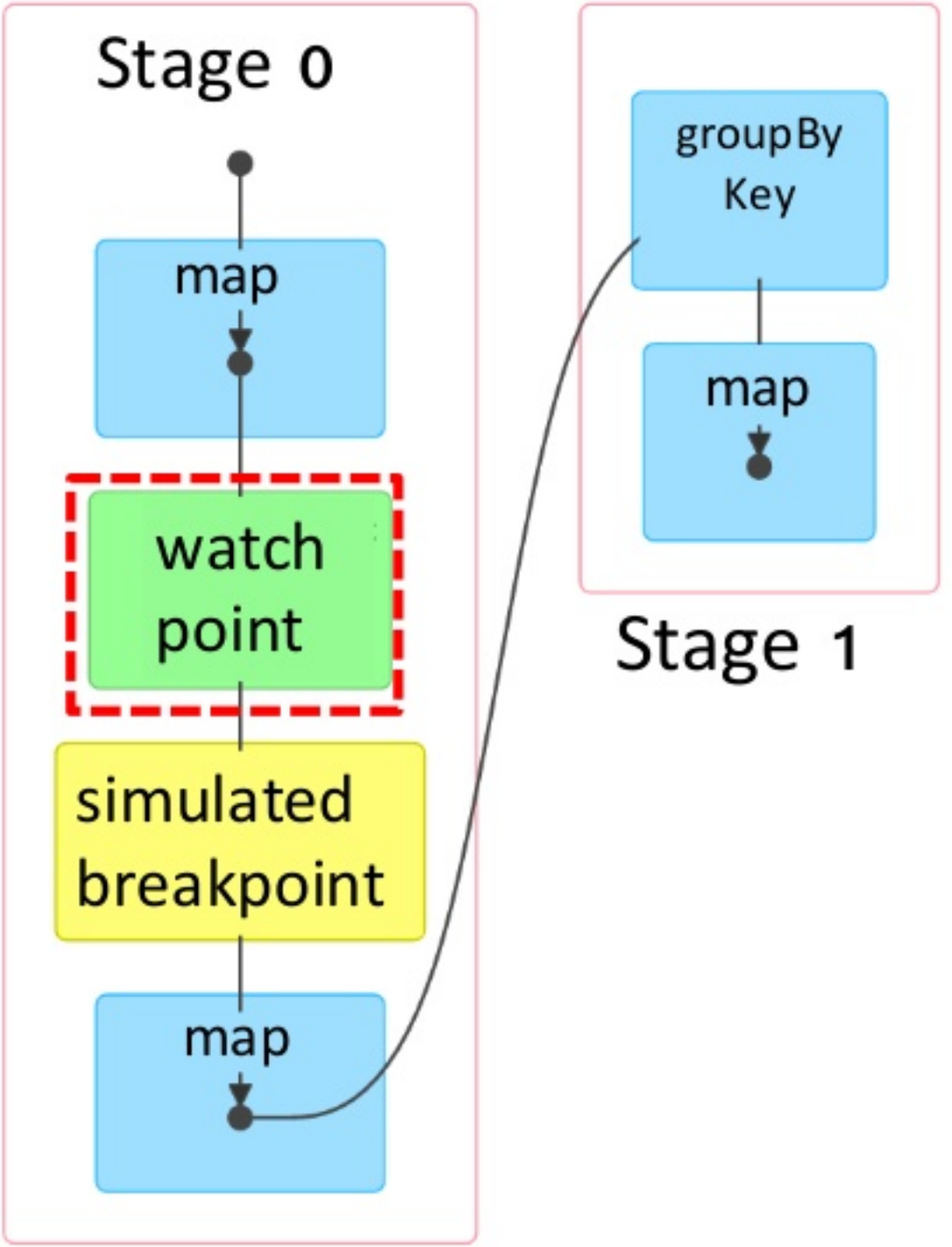


Feature 1: Simulated Breakpoint



Simulated breakpoint enables user to inspect intermediate program state without pausing the computation

Feature 2: On Demand Guarded Watchpoint



Feature 2: On Demand Guarded Watchpoint

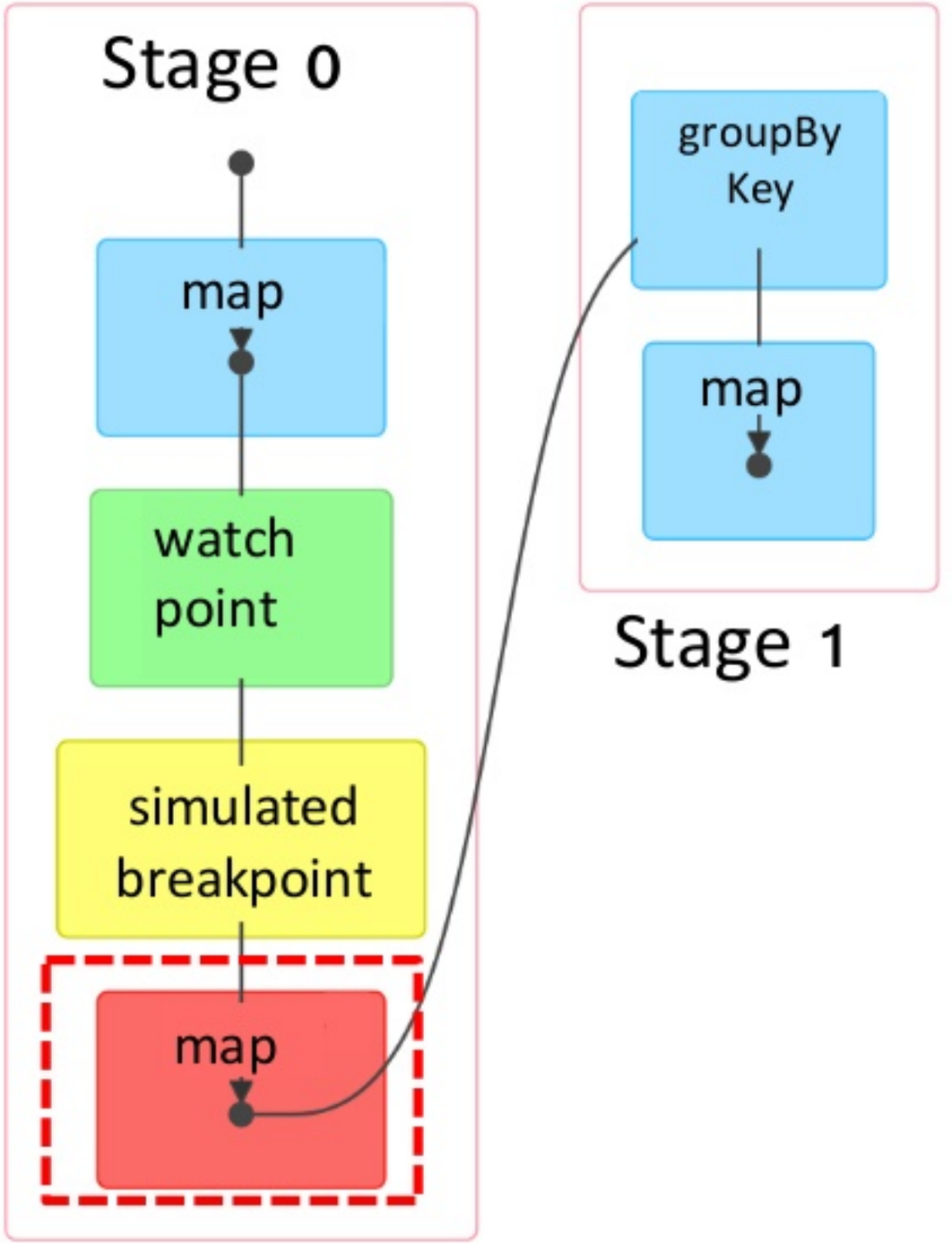
Captured Data Records
1 Timothy 2 21
265 Alan 1 24

```
1  def guard(value:
2  /**Write input types for this watchpoint
3  guard below.For Example : (String, Int) */
4  ): Boolean = {
5  /**Write your guard here**/
6  }
```

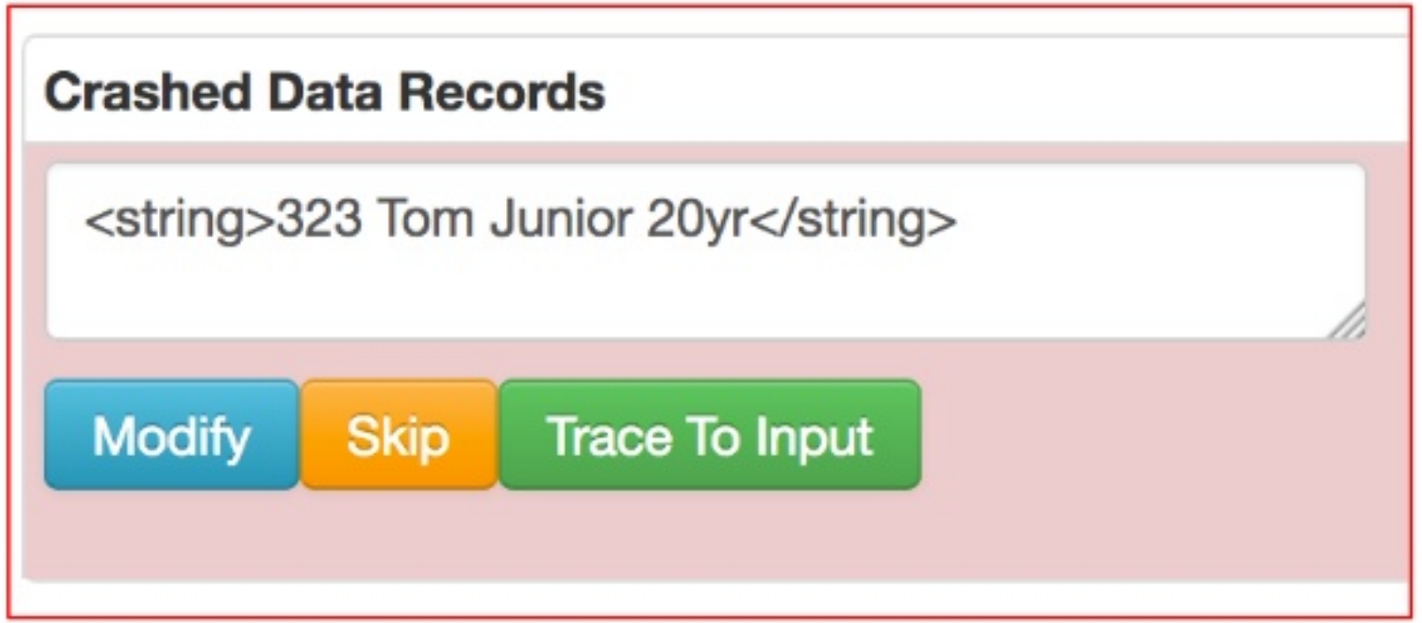
Submit New Guard

A user can inspect intermediate data using a guard and also update it on the fly

Feature 3: Crash Culprit Identification and Remediation

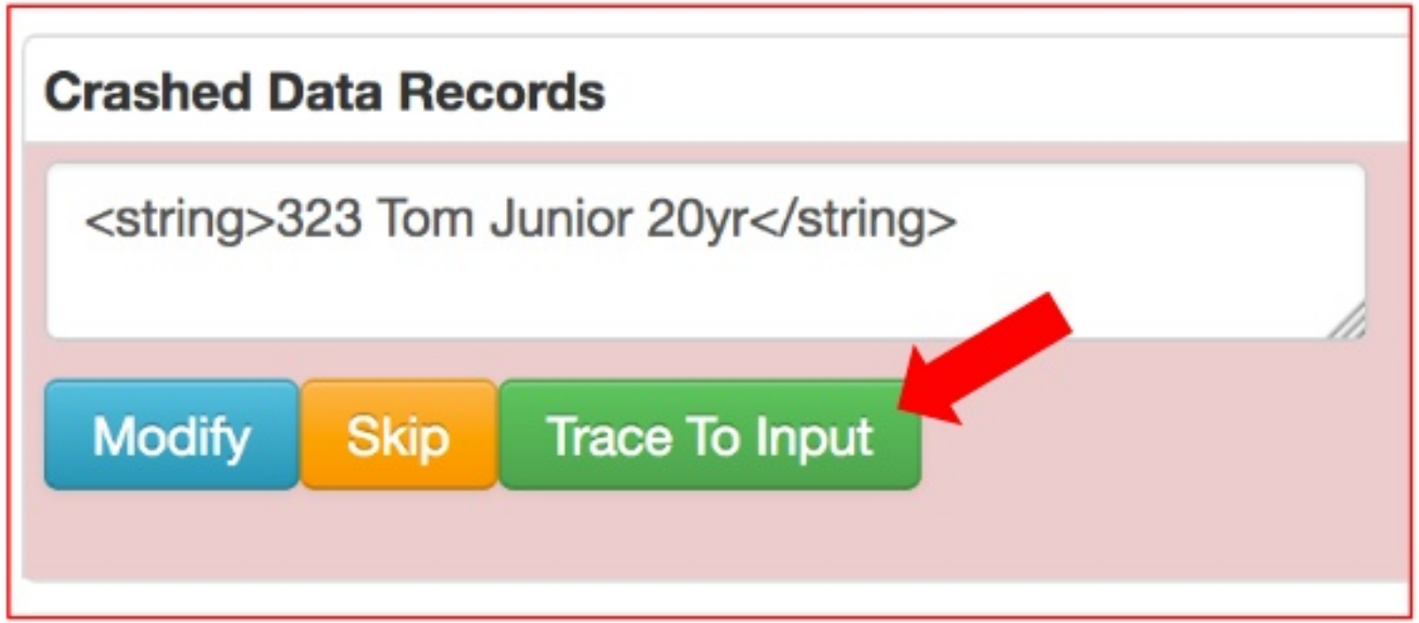


Feature 3: Crash Culprit Identification and Remediation



A user can use BigDebug to identify the crashing records and remediate from the failure

Feature 4: Backward Tracing

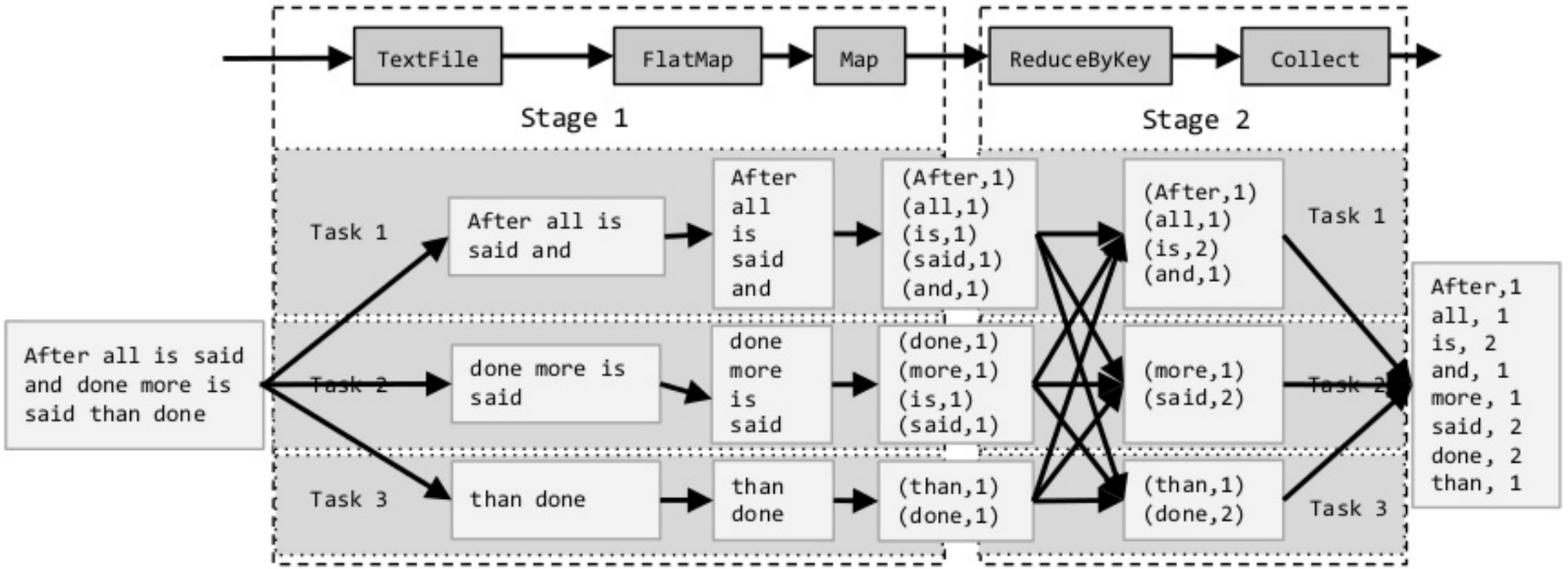


Data Provenance enables users to identify crash inducing inputs records

Feature 5: Automated Fault Localization

Goal: Given a **test function** and set of **failing results**

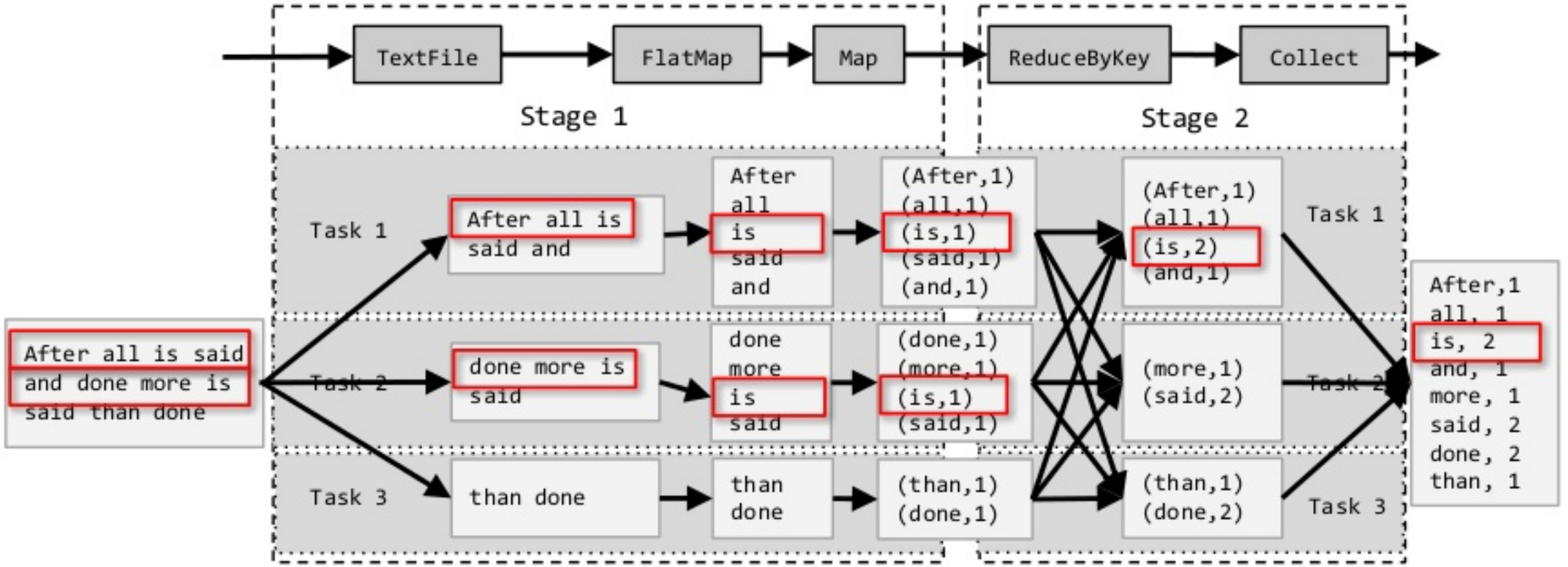
- Identify the minimum set of input records that can reproduce the failure



Feature 5: Automated Fault Localization

Goal: Given a **test function** and set of **failing results**

- Identify the minimum set of input records that can reproduce the failure
- We apply **data provenance** and delta debugging in tandem

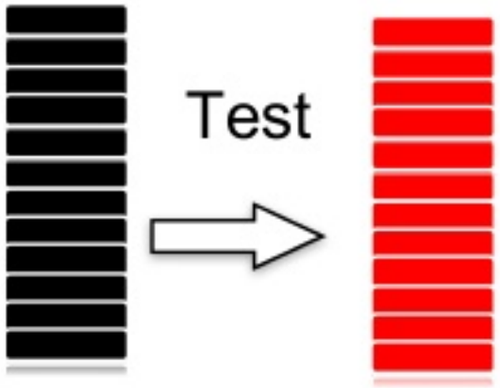


Feature 5: Automated Fault Localization

We apply data provenance and **delta debugging** [Zeller et al.] in tandem

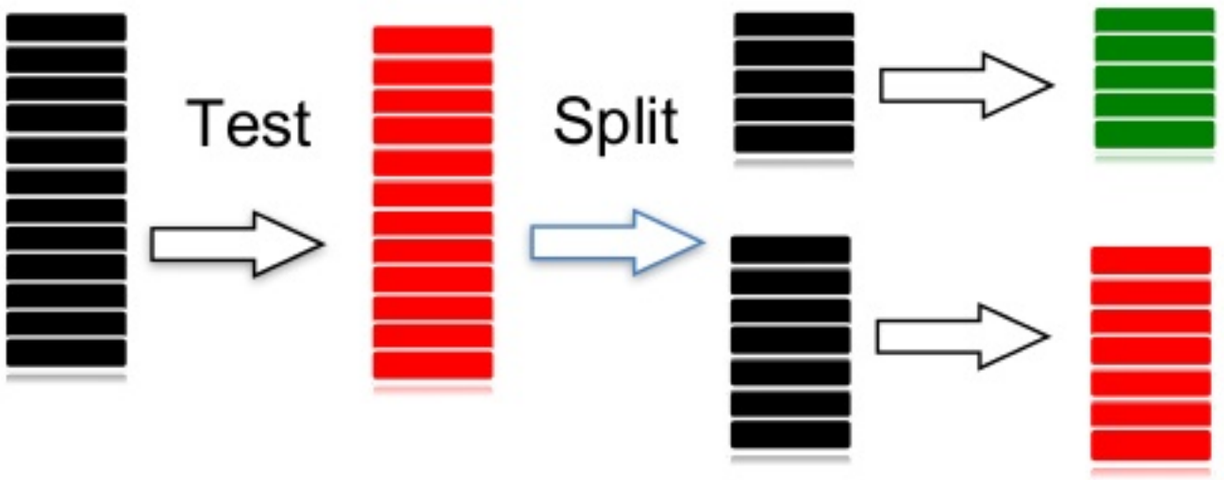
Feature 5: Automated Fault Localization

We apply data provenance and **delta debugging** [Zeller et al.] in tandem



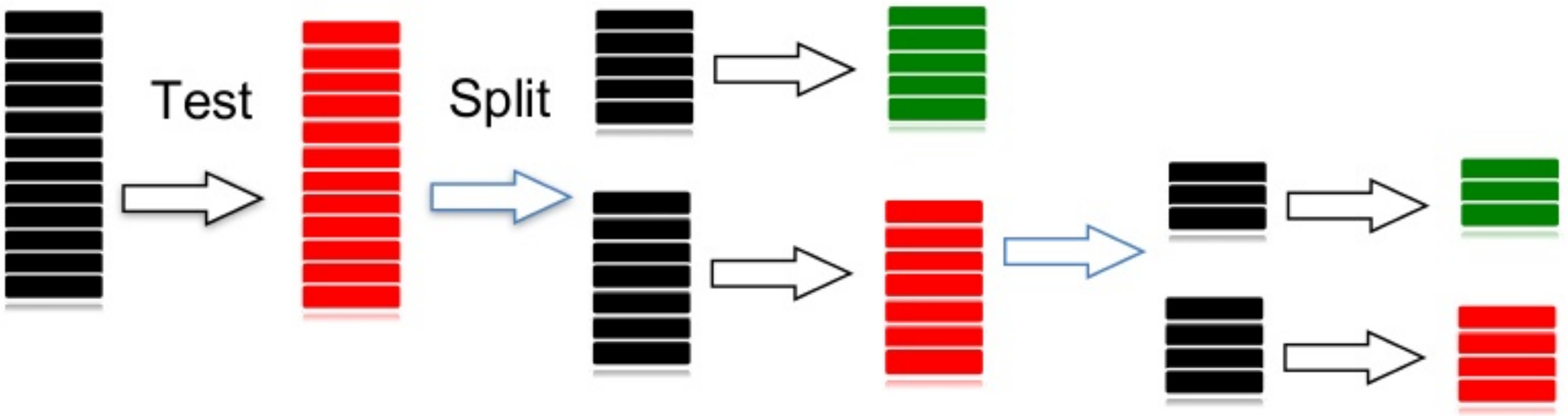
Feature 5: Automated Fault Localization

We apply data provenance and **delta debugging** [Zeller et al.] in tandem



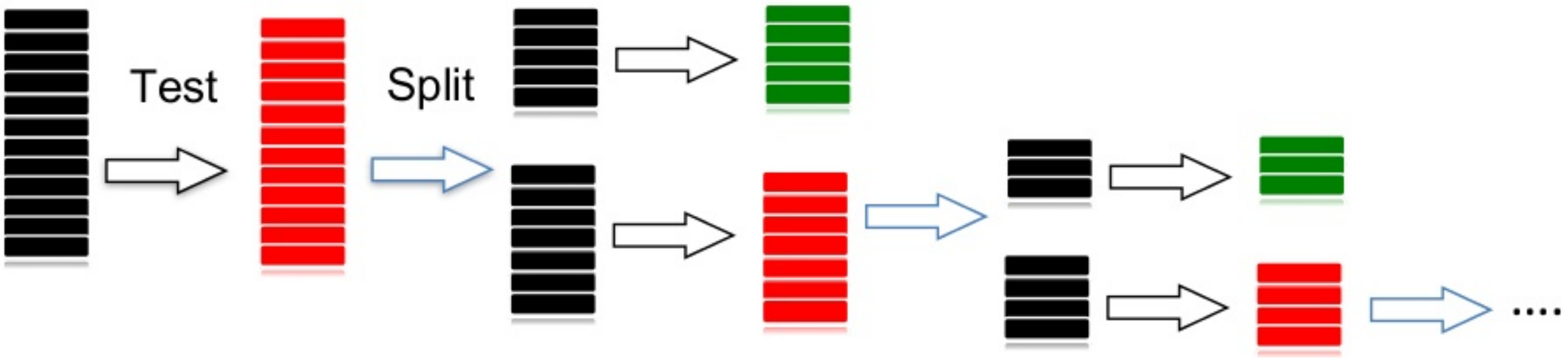
Feature 5: Automated Fault Localization

We apply data provenance and **delta debugging** [Zeller et al.] in tandem



Feature 5: Automated Fault Localization

We apply data provenance and **delta debugging** [Zeller et al.] in tandem



In average BigDebug is able to localize faults within 63% of the original job running time

Demo

Running Example

```
1 Michael Sophomore 03/12/1996
2 Justin  Freshman  05/01/1998
      ..      ..      ..
```

```
val log = "s3n://xcr:wJY@ws/logs/enroll.log"
val text_file = sc.textFile(log)
text_file
.map{line=>(line.split()[2],line.split()[3])}
.map{t => (t._1 , getYears(t._2))}
.groupByKey()
.map(v => (v._1 , average(v._2)))
.collect()
```