BigDebug: Interactive Debugger for Big Data Analytics in Apache Spark

Muhammad Ali Gulzar, Matteo Interlandi, Tyson Condie, Miryung Kim

University of California, Los Angeles

Problem

- Debugging the massive parallel computations that run in today's datacenters is **time consuming and error-prone**.
- The use of cloud computing makes application development feel more like **batch jobs** and the nature of debugging is therefore **post-mortem.**
- Developers are notified of runtime failures or incorrect outputs after many hours of **wasted computing cycles** on the cloud.

Contributions of Our Work

- BigDebug's **simulated breakpoint** enables program state inspection without actually pausing the entire computation.
- Its **on-demand watchpoints** retrieve intermediate data using a dynamic guard predicate.
- BigDebug provides **data provenance** capability, which can help understand how errors propagate through data processing steps.
- BigDebug enables users to change program logic in response to an error at runtime through a **realtime code fix** feature and selectively replay the execution from that step.

Motivating Example

Input Data

```
1 Michael  Sophomore 21
2 Justin   Freshman  19
3 Thomas   Senior    24
        ......
```

Spark Application

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

- Alice intends to compute the average age of all college students in each year (freshman, sophomore, junior, and senior).
- She starts by reading the data into key-value pairs and then groups the records for each category.
- Once she has all related records grouped together, she computes the average and then collects the final results.

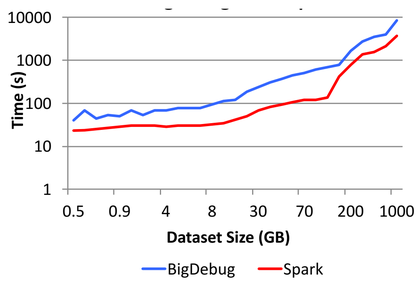
Performance Evaluation and Time Savings

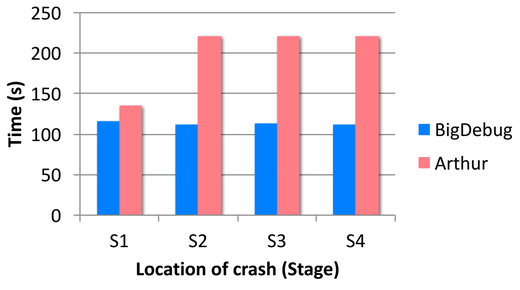
Performance Evaluation

- With the maximum instrumentation setting, BigDebug takes 2.5 times longer than the baseline Spark.
- If we disable the most expensive record-level latency profiling, overhead is less than **34%**, on average.

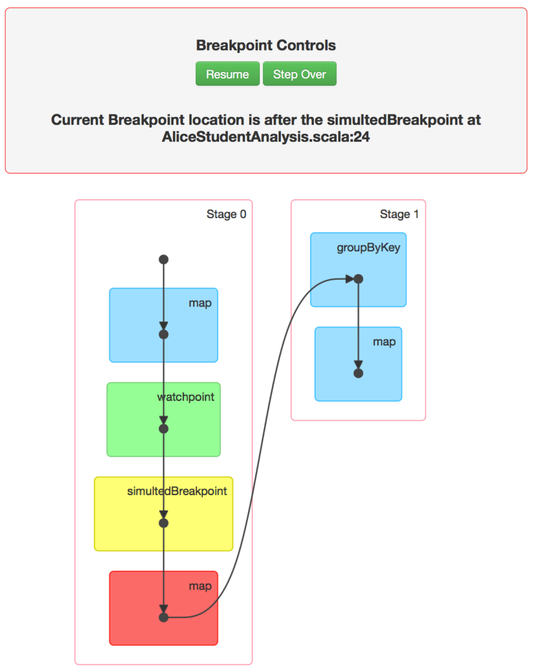
Time Saving

- BigDebug saves upto **100% time saving** over baseline through runtime crash remediation.





Interactive Debugging Primitives



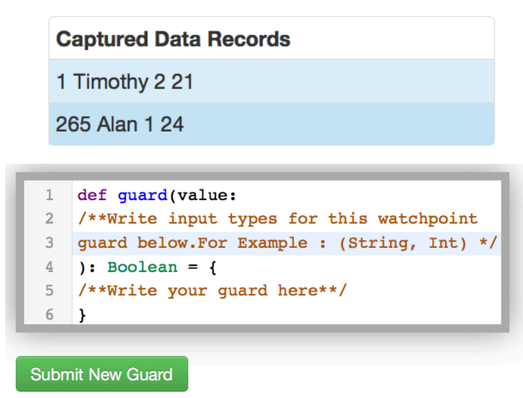
AliceStudentAnalysis.scala

```
10 object AliceStudentAnalysis {
11
12   val COLLEGEYEAR = List("Sophomore" , "Freshman" , "Junior", "Senior")
13   def main(args: Array[String]): Unit = {
14
15     //set up spark configuration
16     val sparkConf = new SparkConf()
17     val bdconf = new BigDebugConfiguration
18     bdconf.setFilePath("/home/ali/work/temp/git/dsbigdebug/spark-lineage/exa
19     //set up spark context
20     val ctx = new SparkContext(sparkConf)
21     ctx.setBigDebugConfiguration(bdconf)
22     //spark program starts here
23     val records = ctx.textFile("/home/ali/Desktop/myfile.txt", 1).
24     simultedBreakpoint(s=> !COLLEGEYEAR.contains(s.split(" ")(2)))
25>    val grade_age_pair = records.map(line => {
26       val list = line.split(" ")
27       (list(2), list(3).toInt)
28     })
29     val average_age_by_grade = grade_age_pair.groupByKey
30       .map(pair => {
31       val itr = pair._2.toIterator
32       var moving_average = 0
33       var num = 1
34       while (itr.hasNext) {
35         moving_average = moving_average + itr.next()
36         num = num + 1
37       }
38       (pair._1, moving_average/num)
39     })
40     val out = average_age_by_grade.collect()
41     out.foreach(println)
42   }
43 }
44
```

- Using this interface, a user can view the DAG of the data flow program. On the right hand side, a user can use the code editor window to see the Spark program in execution.

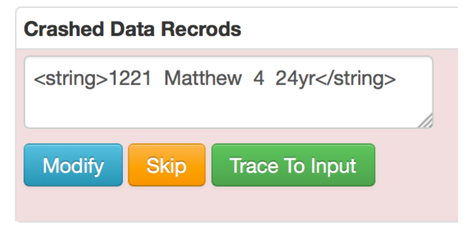
Simulated Breakpoint and On Demand Watchpoint

- When a breakpoint is in place, a program state is regenerated, on-demand, from the last materialization point, while the original process is still running in the background.
- Alice can perform **realtime code fix** using code editor in BigDebug's UI.
- Alice can **resume** and **step-over** instructions using simulated breakpoint controls.
- On-demand guarded watchpoint allows Alice to retrieve intermediate data matching a user-defined **dynamic guard** and transfer the selected data **on demand**.



Crash Culprit Remediation

- BigDebug alerts Alice on the intermediate record responsible for the crash.
- These alerts turn the corresponding transformation node of the DAG to be red.
- When Alice clicks on the red node in the DAG, she is redirected to the crash culprit page where Alice may **skip or modify** the crash inducing intermediate record.



Forward and Backward Tracing

- Alice can invoke the tracing query using "*Trace to Input*" button to perform **step-by-step backward tracing**, showing all intermediate records tracing back to crash-inducing input records.

Fine-Grained Latency Monitoring

- To localize performance anomalies at the record level, BigDebug reports the top *"k"* **straggler records** to the debugger UI.