PerfDebug: Performance Debugging of Computation Skew in Dataflow Systems

Jason Teoh, Muhammad Ali Gulzar, Harry Xu, Miryung Kim
University of California, Los Angeles

UCLA

Problem

Computation Skew: Uneven distribution of *computation* due to interaction between *code* and *data.*

Term
Hello World
Big Data
Debugging
PerfDebug

User-defined function

```
commonDefs = {
  "Hello World": ...,
  "Big Data": ...,
  "Debugging": ...,
  ...
}

if (commonDefs.contains(term)) {
  return commonDefs.get(term)
} else {
  // remote machine lookup
  r = new RedisClient(…)
  return r.get(term)
}
```

Term	Latency
Hello World	2 ms
Big Data	1 ms
Debugging	3 ms
PerfDebug	**442 ms**

Fig. 1: A user-defined function (UDF) that demonstrates computation skew.

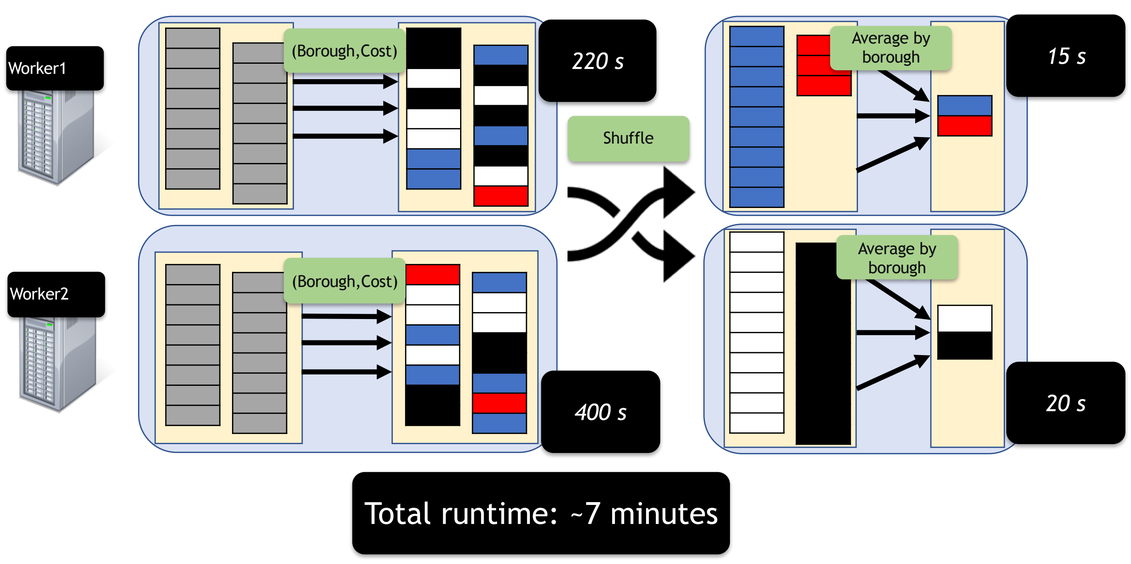


Fig. 2: NYC Taxi Trips program to compute average trip cost per borough, along with task execution times indicating computation skew in the first stage.

Current State of the Art

- Coarse-grained metrics (Spark Web UI) can help *detect* computation skew but cannot find root causes.

Spark 2.1.1 Jobs Stages Storage Environment Executors

Index	ID	Executor ID / Host	Duration	GC Time	Input Size / Records
33	33	8 / 131.179.96.204	1.2 min	7 s	128.0 MB / 17793
34	34	1 / 131.179.96.211	51 s	11 s	128.0 MB / 1
35	35	5/ 131.179.96.212	44s	3 s	128.0 MB / 1
25	25	5 / 131.179.96.212	38 s	2 s	128.0 MB / 33602
36	36	9 / 131.179.96.206	36 s	4 s	128.0 MB / 1
130	130	1 / 131.179.96.211	36 s	9 s	128.0 MB / 33505
37	37	6 / 131.179.96.203	35s	4 s	128.0 MB / 1
22	22	3 / 131.179.96.209	35 s	2 s	128.0 MB / 33564

Fig. 4: Example Spark Web UI Task metrics, with a skewed task outlined in red.

- Data provenance systems such as Titian [1] trace record flow and provide root cause analysis, but do not analyze performance.

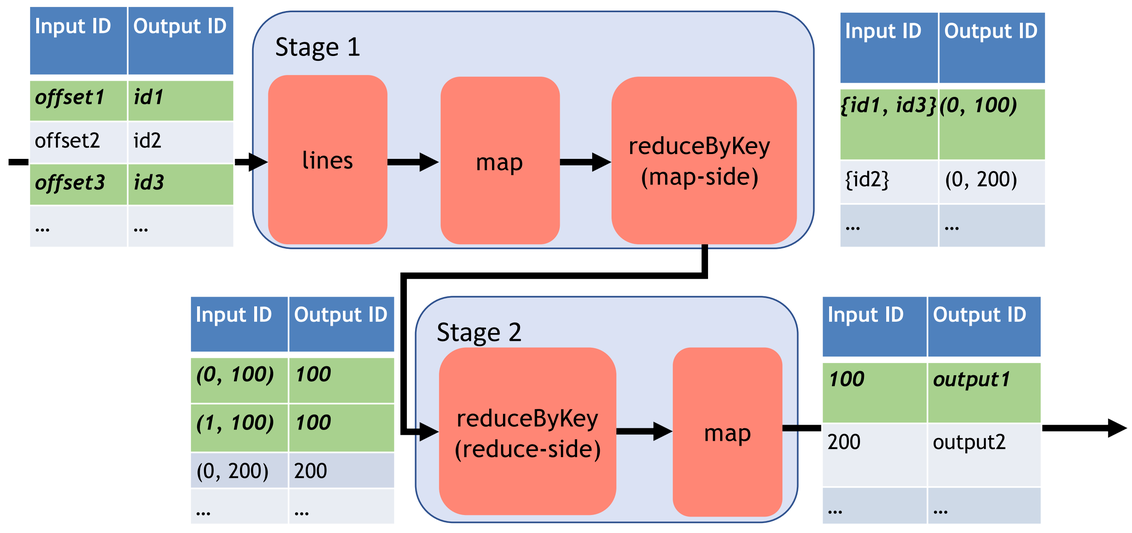
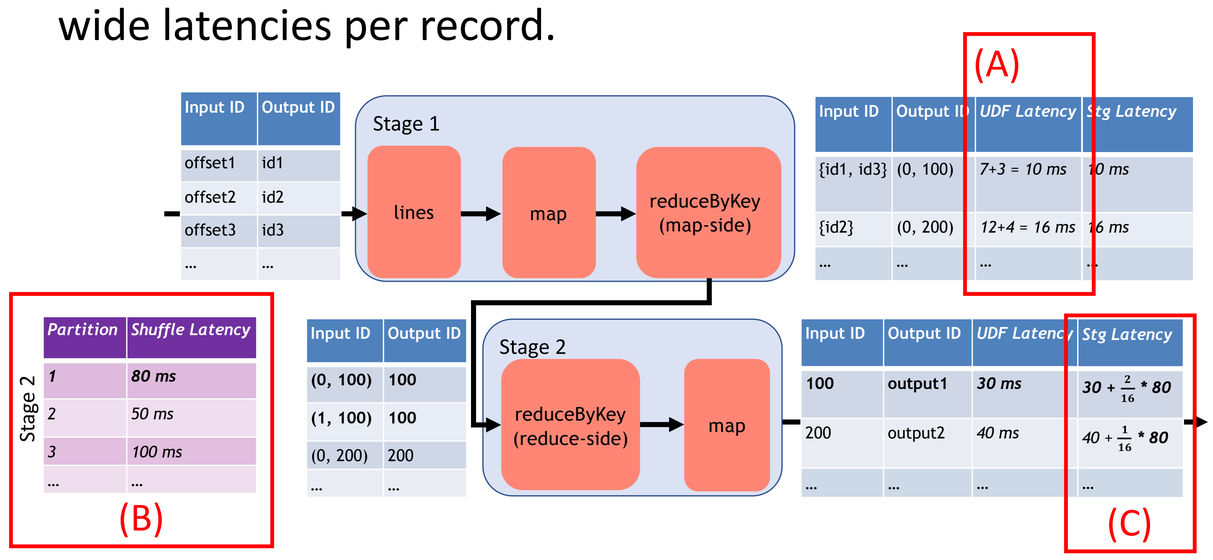


Fig. 5: Example of Titian data provenance and input tracing *output1 (*green*).*

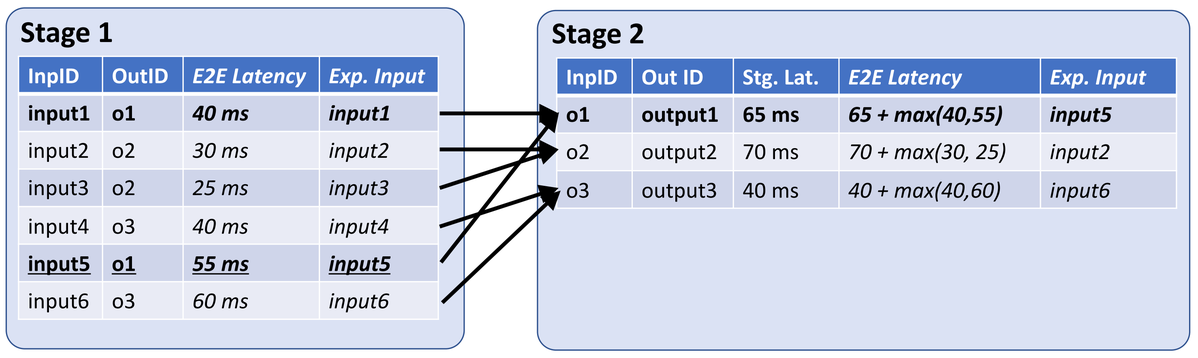
[1] Titian: Data Provenance Support in Spark. Matteo Interlandi, Kshitij Shah, Sai Tetali, Muhammad Gulzar, Seunghyun Yoo, Miryung Kim, Todd Millstein, Tyson Condie. PVLDB Volume 9 Issue 3). Pages 216-227

Approach

- **Detection**: Monitor execution through coarse-grained SparkListener metrics. If computation skew is detected, re-execute in debugging mode.
- **Data Provenance**: Rerun program using PerfDebug's extension of Titian data provenance.
- **Latency Instrumentation**: Collect (A) UDF latencies per record and (B) shuffle latencies to compute (C) stage-wide latencies per record.



- **Latency Propagation + Input Tracking**: Forward-propagate end-to-end latencies by summing along the longest latency path, while retaining the start (program input) for each path.



Evaluation

- PerfDebug identifies many computation skew causes such as data skew, data quality, and expensive UDFs.
- Guided performance fixes yield up to **16X improvement**.

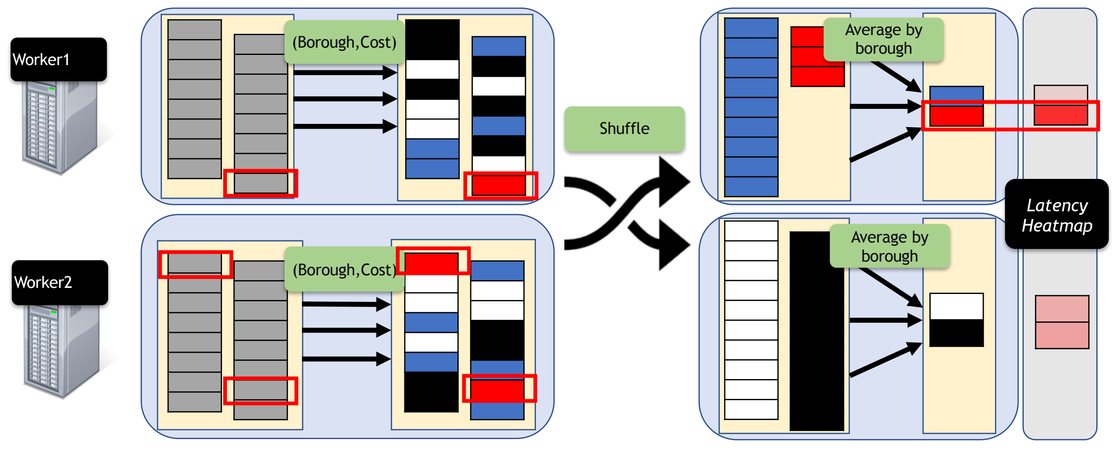


Fig. 6: NYC Taxi Trip case study and how PerfDebug was able to identify an expensive input subset by analyzing output record latencies.

Benchmark	Accuracy	Precision Improvement	Overhead
Movie Ratings	100%	2102X	1.04X
College Students	100%	1250000X	1.39X
Weather Analysis	100%	294X	1.48X
Average	100%	417465X	1.30X

Table 1: Evaluation results on three benchmarks compared to Titian. The ~30% overhead is primarily due to persistent storage for post-mortem debugging.