Big data platform (Spark) performance acceleration

*Mentors: Tony Tan, Ning Wu, Yong Wang and **Theo Gkountouvas***

By:

Grishma Atul Thakkar

Virat Goradia

Nipun Midha

Baoshu Brady Qi

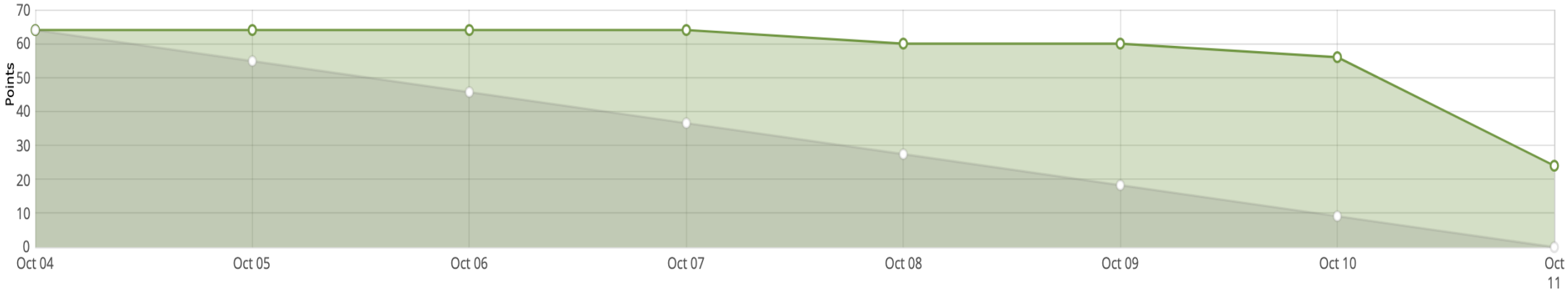
Sprint Goals

- Understand existing spark code
- Find the appropriate data set
- Start implementing the N-Way merge algorithm
- Research on ways to implement the N-Way merge

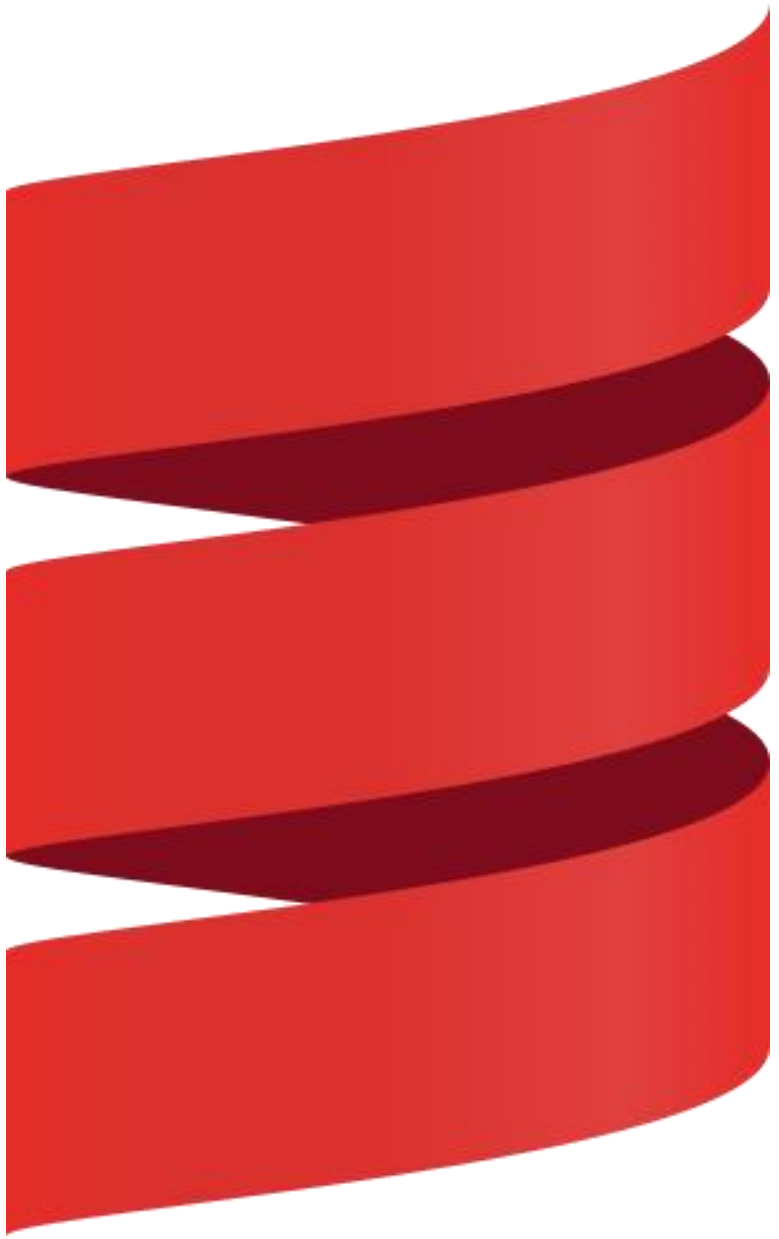
Burndown Chart

DEMO 2 NEU6620-BIG DATA PLATFORM (SP... 04 OCT 2019-11 OCT 2019

63% 64 total points 40 completed points 0 open tasks 0 closed tasks 0 iocaine doses



Spike

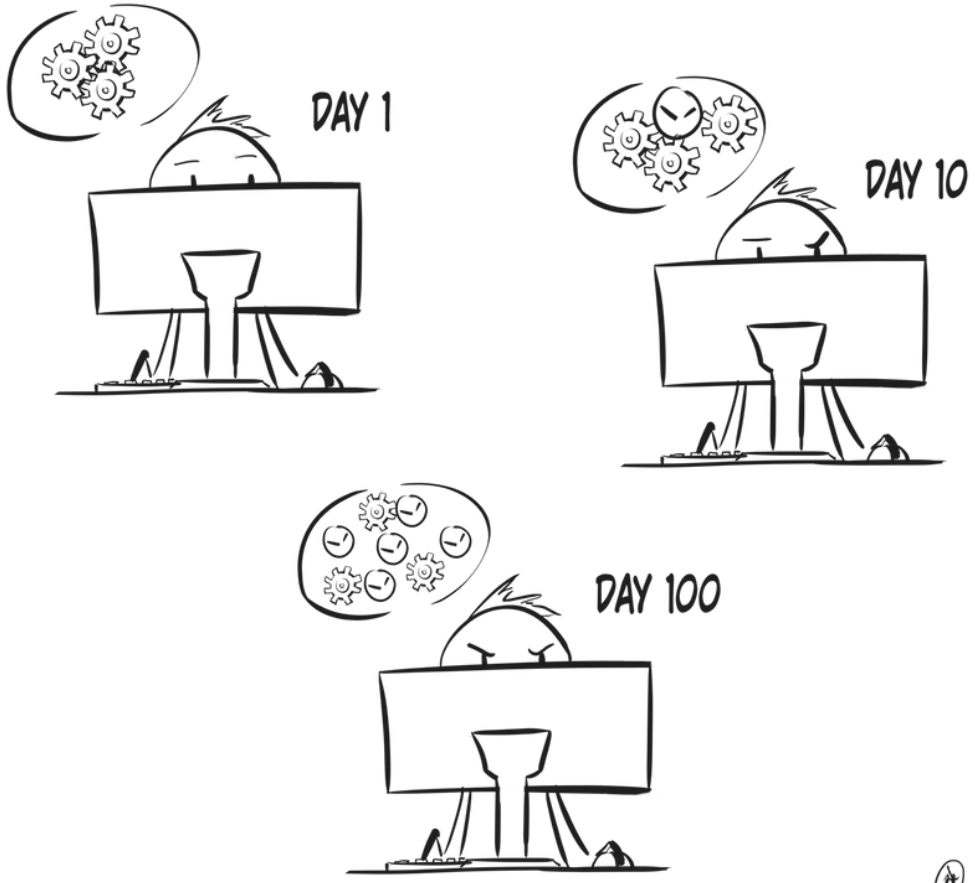


Building Spark Code

```
[INFO]
[INFO] --- maven-source-plugin:3.1.0:test-jar-no-fork (create-source-jar) @ spark-avro_2.12 ---
[INFO] Building jar: C:\Users\virat\Desktop\FCC\spark\external\avro\target\spark-avro_2.12-3.0.0-SNAPSHOT-test-sources.jar
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for Spark Project Parent POM 3.0.0-SNAPSHOT:
[INFO]
[INFO] Spark Project Parent POM ........................... SUCCESS [ 28.330 s]
[INFO] Spark Project Tags ................................. SUCCESS [ 14.683 s]
[INFO] Spark Project Sketch ............................... SUCCESS [ 12.568 s]
[INFO] Spark Project Local DB ............................. SUCCESS [  4.529 s]
[INFO] Spark Project Networking ........................... SUCCESS [ 11.149 s]
[INFO] Spark Project Shuffle Streaming Service ............ SUCCESS [  4.290 s]
[INFO] Spark Project Unsafe ............................... SUCCESS [ 18.983 s]
[INFO] Spark Project Launcher ............................. SUCCESS [  8.125 s]
[INFO] Spark Project Core ................................. SUCCESS [04:51 min]
[INFO] Spark Project ML Local Library ..................... SUCCESS [ 51.199 s]
[INFO] Spark Project GraphX ............................... SUCCESS [01:09 min]
[INFO] Spark Project Streaming ............................ SUCCESS [01:42 min]
[INFO] Spark Project Catalyst ............................. SUCCESS [04:56 min]
[INFO] Spark Project SQL .................................. SUCCESS [05:40 min]
[INFO] Spark Project ML Library ........................... SUCCESS [03:56 min]
[INFO] Spark Project Tools ................................ SUCCESS [ 12.725 s]
[INFO] Spark Project Hive ................................. SUCCESS [03:04 min]
[INFO] Spark Project Graph API ............................ SUCCESS [  3.057 s]
[INFO] Spark Project Cypher ............................... SUCCESS [  4.654 s]
[INFO] Spark Project Graph ................................ SUCCESS [  3.688 s]
[INFO] Spark Project REPL ................................. SUCCESS [ 36.138 s]
[INFO] Spark Project Assembly ............................. SUCCESS [  6.209 s]
[INFO] Kafka 0.10+ Token Provider for Streaming ........... SUCCESS [ 49.430 s]
[INFO] Spark Integration for Kafka 0.10 ................... SUCCESS [01:23 min]
[INFO] Kafka 0.10+ Source for Structured Streaming ........ SUCCESS [02:19 min]
[INFO] Spark Project Examples ............................. SUCCESS [01:40 min]
[INFO] Spark Integration for Kafka 0.10 Assembly .......... SUCCESS [ 10.549 s]
[INFO] Spark Avro ......................................... SUCCESS [02:11 min]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  37:42 min
[INFO] Finished at: 2019-10-10T18:53:32-04:00
[INFO] ------------------------------------------------------------------------
```

Spark Codebase

839,842 **lines of code**



Demo

	follower	followee
0	1	11553
1	1	8762940
2	1	8762941
3	1	688136
4	1	8762942

m4.large
4 Cores, 8GiB memory,
32GiB EBS storage

MAXFILTER = 15,000

```
val textFile = sc.textFile(args(0))
val MAXFILTER = Integer.valueOf(args(2))

val textContent = textFile.map(line => (line.split( regex = ",")(0), line.split( regex = ",")(1)))
val first_path = textContent.filter(
  (x) =>  Integer.valueOf(x._1) < MAXFILTER  && Integer.valueOf(x._2) < MAXFILTER)

val second_path = first_path.map(x => (x._2, x._1))
first_path.join(second_path).saveAsTextFile(args(1))
```

Demo

Summary metrics for 20 completed tasks

Metric ▲	Min	25th percentile	Median	75th percentile	Max
Duration	36 s	1.3 min	1.6 min	2.0 min	2.8 min
GC time	2 s	4 s	4 s	5 s	6 s
Output (size / records)	369.1 MiB / 18,314,343	845.7 MiB / 43,143,454	1.2 GiB / 62,595,858	1.5 GiB / 75,866,051	2.6 GiB / 135,073,645
Result serialization time			1 ms	4 ms	19 ms
Shuffle read (size / records)	585.8 KiB / 88,302	647.8 KiB / 98,769	748.3 KiB / 115,884	826.5 KiB / 130,788	1,006.0 KiB / 159,654
Shuffle remote reads	489.6 KiB	575.2 KiB	717.1 KiB	748.3 KiB	871.5 KiB
Task deserialization time	0.1 s	0.2 s	0.2 s	0.3 s	0.4 s

Demo

```scala
val textFile = sc.textFile(args(0))
val MAXFILTER = Integer.valueOf(args(2))
val textContent = textFile.map(line => (line.split( regex = ",")(0), line.split( regex = ",")(1)))

val first_path:RDD[(String, String)] = textContent.filter(
  (x) =>  Integer.valueOf(x._1) < MAXFILTER  && Integer.valueOf(x._2) < MAXFILTER)

val second_path:RDD[(String, String)] = first_path.map(x => (x._2, x._1))
val path2 = first_path.join(second_path).map {
  case (mid, (end, start)) => (end, (start, mid))
}

path2.join(second_path).saveAsTextFile(args(1))
```

Succeeded tasks	Output	Shuffle read	Shuffle spill (memory)	Shuffle spill (disk)
5	57.5 GiB	223.5 MiB	3.0 GiB	194.7 MiB
5	106.8 GiB	218.6 MiB	2.9 GiB	151.1 MiB
5	83.1 GiB	222.2 MiB	1.8 GiB	98.5 MiB
5	70.3 GiB	216.7 MiB	1.8 GiB	118.2 MiB

Demo

Summary metrics for 20 completed tasks

Metric ▲	Min	25th percentile	Median	75th percentile	Max
Duration	24 min	32 min	46 min	55 min	1.6 h
GC time	4.0 min	6.7 min	10 min	16 min	28 min
Output (size / records)	6.1 GiB / 242,721,205	11.1 GiB / 449,848,182	12.5 GiB / 499,215,490	18.7 GiB / 754,456,206	59.8 GiB / 2,387,116,509
Result serialization time				1 ms	24 ms
Shuffle read (size / records)	41.8 MiB / 8,036,627	43.0 MiB / 8,270,179	44.2 MiB / 8,474,130	45.4 MiB / 8,696,210	46.7 MiB / 8,931,070
Shuffle remote reads	24.4 MiB	31.2 MiB	33.8 MiB	38.8 MiB	41.1 MiB
Shuffle spill (disk)	0.0 B	27.2 MiB	32.0 MiB	41.3 MiB	45.5 MiB
Shuffle spill (memory)	0.0 B	597.8 MiB	597.8 MiB	597.8 MiB	636.5 MiB
Task deserialization time	64 ms	75 ms	84 ms	0.1 s	0.1 s

Total time across all tasks: 14.9 h

Locality level summary: Process local: 20

Output (size / records): 317.7 GiB / 12,720,501,154

Shuffle read (size / records): 881.0 MiB / 169,081,924

Shuffle spill (memory): 9.4 GiB

Shuffle spill (disk): 562.4 MiB

MAXFILTER = 30,000

Terminated manually, after the cluster running about 3h30min.

- **Possible solutions in the future on how to get spillages:**
- Multiple join actions on relatively small dataset
- Use machines with smaller memory
- Alternatively, manually set smaller executor memory, shuffle memory, etc.
    - --spark.executor.memory

Output	Shuffle read	Shuffle spill (memory)	Shuffle spill (disk)
10.3 GiB	431.7 MiB	6.6 GiB	364.9 MiB
10.7 GiB	424.5 MiB	7.2 GiB	336.5 MiB
14.6 GiB	420.3 MiB	5.9 GiB	391.1 MiB
10.6 GiB	413.6 MiB	5.8 GiB	363.7 MiB

Important Files

ShuffleManager

Register Shuffle

Unregister Shuffle

Get ShuffleReader

Get ShuffleWriter

Stop the shuffle system

ShuffleMapStage

- ShuffleMapStage is an **intermediate stage** that produces data for other stage(s). It writes **map output files** for a shuffle.
- When all map outputs are available, the ShuffleMapStage is considered **available** (or **ready**).
- ShuffleMapStage uses **outputLocs** and **_numAvailableOutputs** internal registries to track how many shuffle map outputs are available.
- outputLocs gives us information of the MapStatus (Ready or not).
- NumAvailableOutputs gives us the information as to how many "ready" outputs are available, which would help us in the N-way merge algorithm.

ShuffleWriteMetrics

ShuffleWriteMetrics tracks the following metrics:

A) Shuffle bytes written

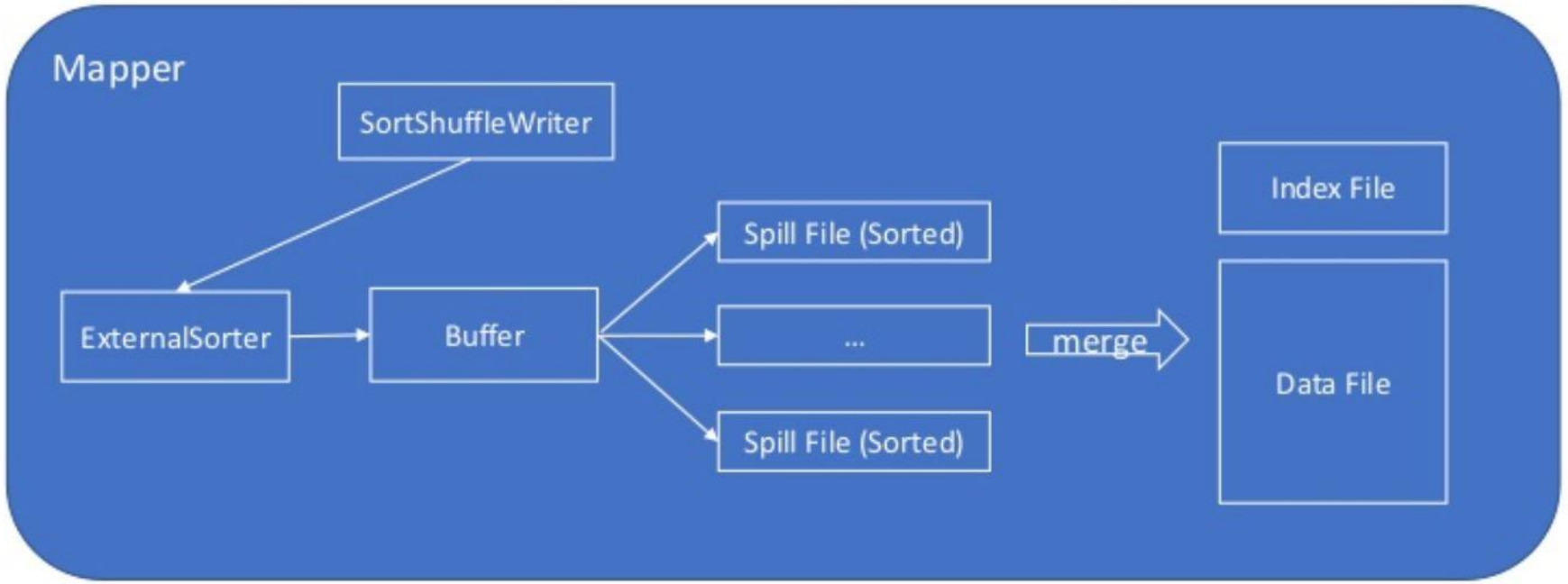
B) Shuffle Write time

C) Shuffle records written

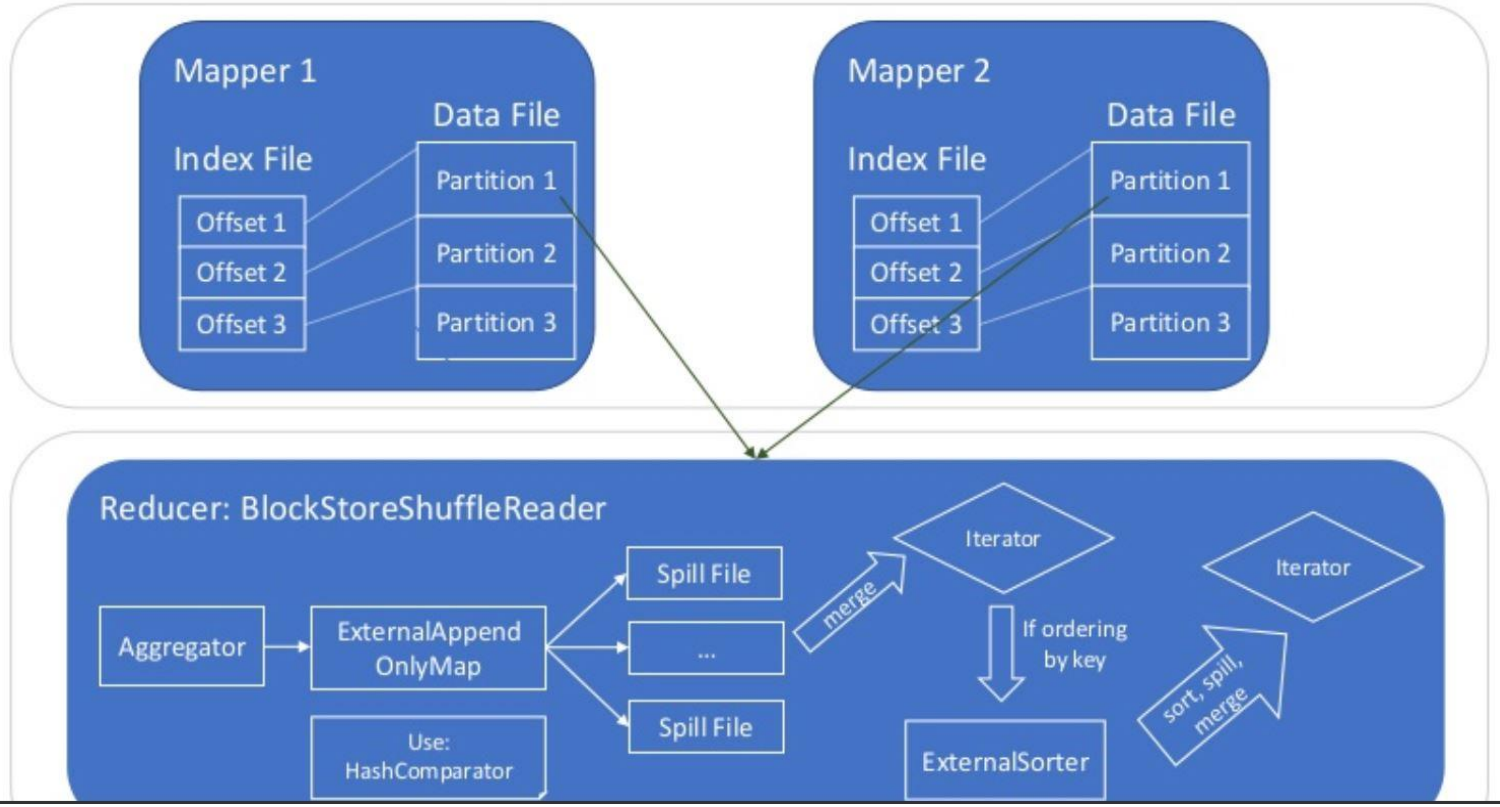
Expected observation:

- Shuffle bytes written must remain the same.
- Shuffle write time must decrease.
- Shuffle records written must decrease.

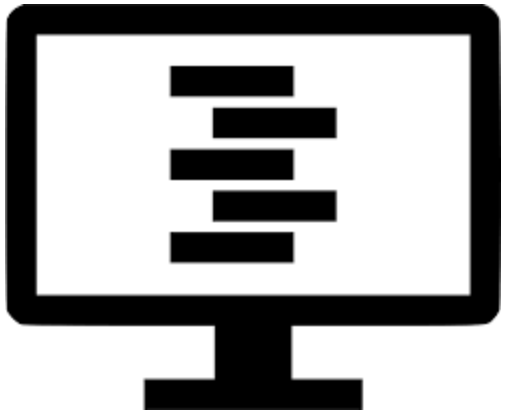
SortShuffleWriter

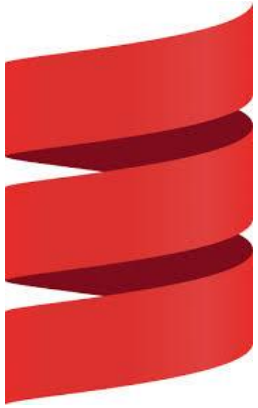


BlockStoreShuffleReader



Challenges





Next Sprint Goals

- Understand existing spark code
- Find the appropriate benchmarks
- Research on ways to implement the N-Way merge
- Start implementing the N-Way merge algorithm

Any Questions?

Thank You!