livelessons
video instruction from technology experts

Lesson 4: Spark Internals

4.10 Tuning Your Spark Application

Details for Job 16

Status: SUCCEEDED
Job Group: Airline Data
Completed Stages: 2

▸ Event Timeline
▸ DAG Visualization

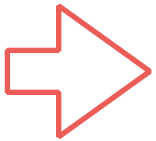
Completed Stages (2)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
38	no filter sortBy at <ipython-input-27-c58242d90558>:9	2015/08/13 15:53:31	30 s	11/11			117.7 MB	
37	no filter groupByKey at <ipython-input-27-c58242d90558>:7	2015/08/13 15:39:24	14 min	11/11	256.3 MB			117.7 MB

Performance Tuning

Monitoring

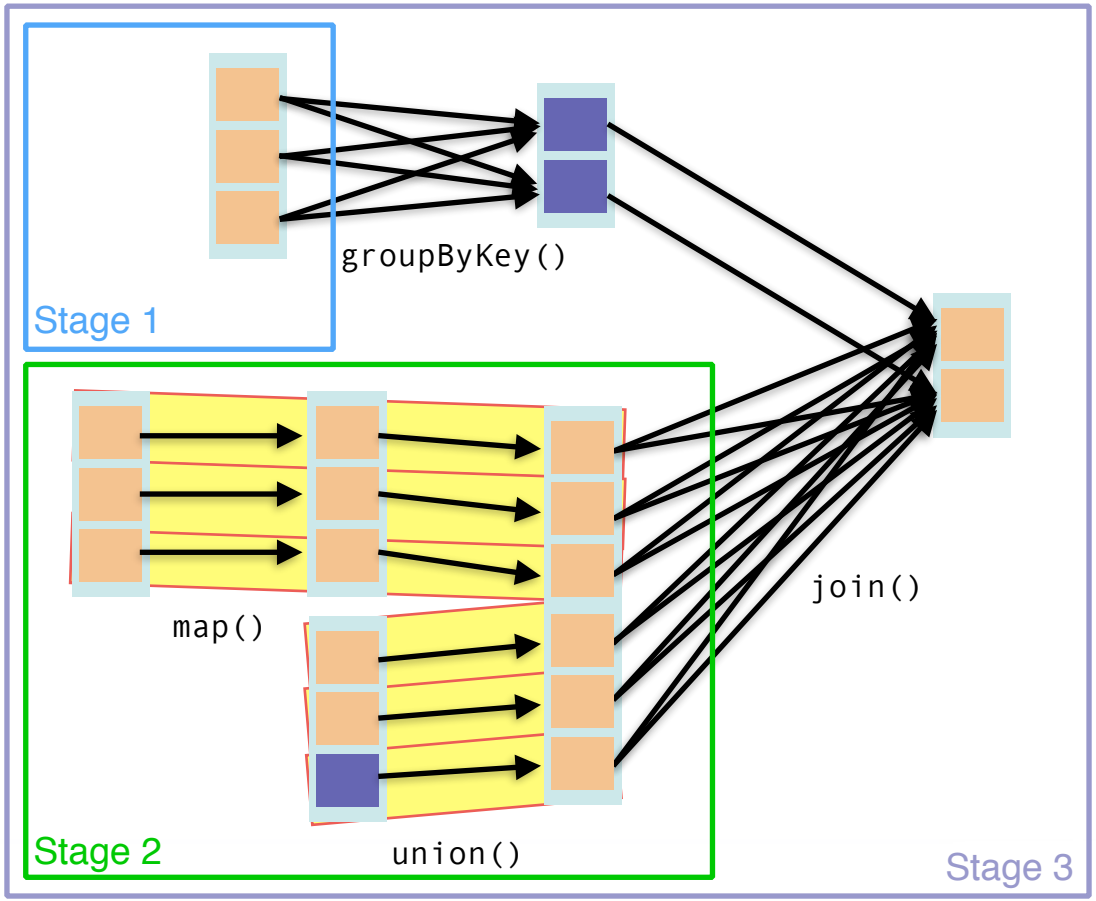
- Web UI (application)
- History Server (application)
- Ganglia (infrastructure)
- jstack, jmap, jstat (JVM)



Optimization

- Data (serialization + locality)
- Application
  - data structures
  - caching
  - broadcasting
  - shuffle
- Framework (parallelism + memory + GC)

Remember: Stages and Shuffles



What Spark gives us

- Pipelining of transformation with narrow dependencies
- Data locality to limit data transfer over the network
- Truncated DAG from cached RDD (or persisted from shuffle)

What we need to be aware of

- Shuffles are a very expensive process
- Proper partitioning can increase parallelism
- Still…quantity of data transferred and operated on

Make data small as quick as possible

- `aggregateByKey()`
- `filter()`

Make data small as quick as possible

Spark 1.4.1 | Jobs | Stages | Storage | Environment | Executors | **Performance Tuning** application UI

Spark Jobs (?)

Total Uptime: 2.3 h
Scheduling Mode: FIFO
Completed Jobs: 22

▸ Event Timeline

Completed Jobs (22)

Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
21 (Airline Data)	filtered first runJob at PythonRDD.scala:366	2015/08/13 16:08:48	1 s	2/2 (1 skipped)	12/12 (11 skipped)
20 (Airline Data)	filtered first sortBy at <ipython-input-28-824c2b202e95>:8	2015/08/13 16:08:47	0.8 s	1/1 (1 skipped)	11/11 (11 skipped)
19 (Airline Data)	filtered first sortBy at <ipython-input-28-824c2b202e95>:8	2015/08/13 15:55:04	14 min	2/2	22/22 (1 failed)
18 (Airline Data)	no filter runJob at PythonRDD.scala:366	2015/08/13 15:54:34	30 s	2/2 (1 skipped)	12/12 (11 skipped)
17 (Airline Data)	no filter sortBy at <ipython-input-27-c58242d90558>:9	2015/08/13 15:54:02	32 s	1/1 (1 skipped)	11/11 (11 skipped)
16 (Airline Data)	no filter sortBy at <ipython-input-27-c58242d90558>:9	2015/08/13 15:39:24	15 min	2/2	22/22

Make data small as quick as possible

Spark 1.4.1 Jobs Stages Storage Environment Executors **Performance Tuning** application UI

Details for Stage 37 (Attempt 0)

Total Time Across All Tasks: 28 min
Input Size / Records 256.3 MB / 5113194
Shuffle Write: 117.7 MB / 330

- DAG Visualization
- Show Additional Metrics
- Event Timeline

Summary Metrics for 11 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	1 s	2.0 min	2.5 min	3.2 min	3.9 min
GC Time	0 ms	10 ms	13 ms	25 ms	33 ms
Input Size / Records	139.0 B / 1	24.5 MB / 488007	25.2 MB / 505219	26.3 MB / 526934	27.2 MB / 545132
Shuffle Write Size / Records	0.0 B / 0	11.1 MB / 33	11.4 MB / 33	12.4 MB / 33	13.0 MB / 33

Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input Size / Records	Shuffle Write Size / Records
0	10.25.111.149:60908	14 min	6	0	6	154.7 MB / 3086423	71.1 MB / 198
1	10.25.111.149:60907	13 min	5	0	5	101.6 MB / 2026771	46.7 MB / 132

Make data small as quick as possible

Spark 1.4.1 Jobs Stages Storage Environment Executors **Performance Tuning** application UI

Details for Stage 44 (Attempt 0)

Total Time Across All Tasks: 27 min
Input Size / Records: 256.3 MB / 5113194
Shuffle Write: 10.9 MB / 302

- DAG Visualization
- Show Additional Metrics
- Event Timeline

Summary Metrics for 12 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	0.4 s	2.0 min	2.1 min	3.0 min	3.5 min
GC Time	0 ms	5 ms	7 ms	8 ms	10 ms
Input Size / Records	139.0 B / 1	24.5 MB / 488007	25.2 MB / 505219	26.3 MB / 526934	27.2 MB / 545132
Shuffle Write Size / Records	0.0 B / 0	1057.5 KB / 28	1086.8 KB / 31	1163.9 KB / 32	1225.7 KB / 33

Aggregated Metrics by Executor

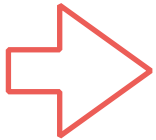
Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input Size / Records	Shuffle Write Size / Records
0	10.25.111.149:60908	14 min	5	1	4	103.3 MB / 2055157	4.4 MB / 116
1	10.25.111.149:60907	13 min	7	0	7	153.0 MB / 3058037	6.6 MB / 186

What we need to be aware of

- Input size evenly distributed on executors?
- Task time evenly distributed?
- If yes, maybe you can benefit from increased parallelism (i.e. get more machines)

Avoiding Shuffling

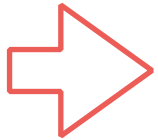
`partitionBy()`



- `join()`
- `reduceByKey()`
- `sortByKey()`

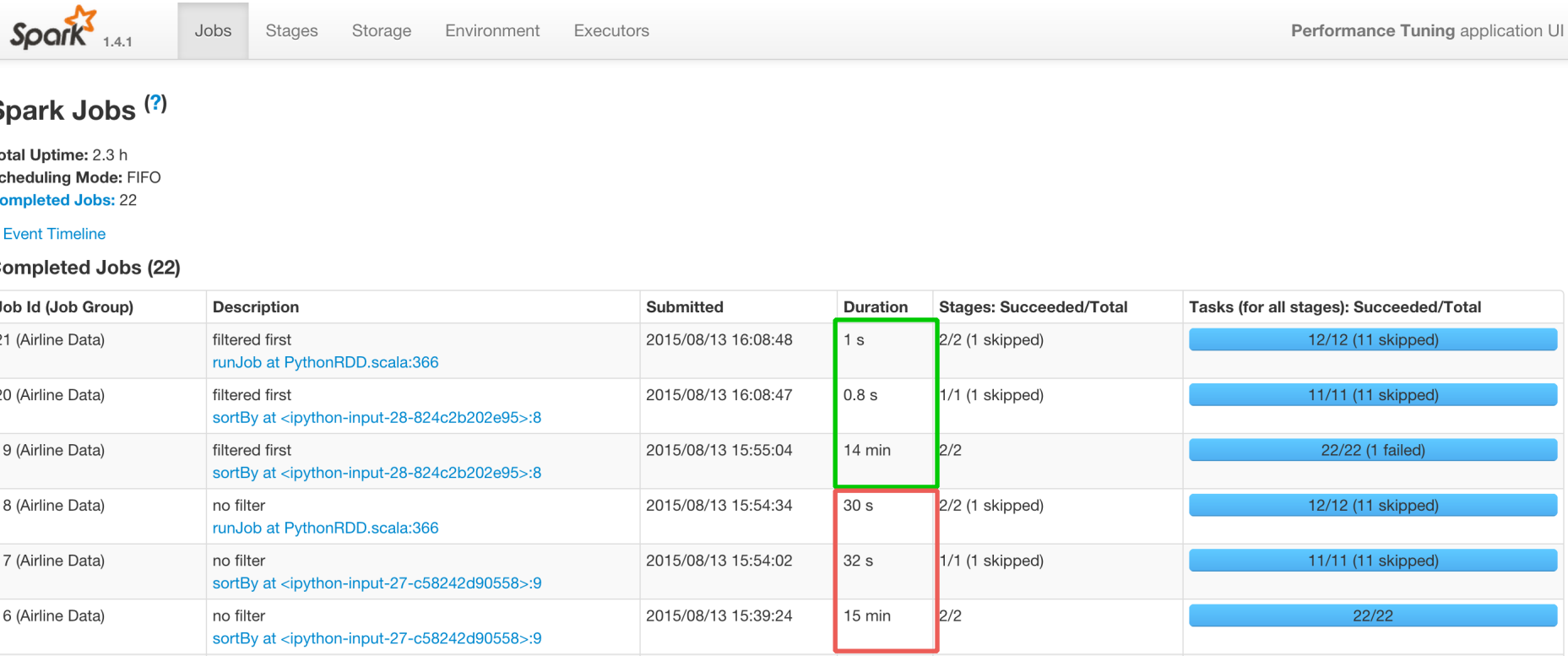
prepared data for key operations with hash-partition + `persist()`

Limit Shuffling

`groupByKey().mapValues()`  `reduceByKey()`

performs "map-side" reduce before shuffle

Limit Shuffling



Limit Shuffling

Spark 1.4.1 | Jobs | Stages | Storage | Environment | Executors | **Performance Tuning** application UI

Spark Jobs (?)

Total Uptime: 3.0 h
Scheduling Mode: FIFO
Active Jobs: 1
Completed Jobs: 34
Failed Jobs: 2

▸ Event Timeline

Active Jobs (1)

Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
36 (Airline Data -- filtered)	sortBy at <ipython-input-44-a8444b85785e>:9	2015/08/13 18:31:48	5.7 min	0/2	5/22

Completed Jobs (34)

Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
35 (Airline Data -- filtered)	reduceByKey + filtered runJob at PythonRDD.scala:366	2015/08/13 18:31:48	0.2 s	2/2 (1 skipped)	12/12 (11 skipped)
34 (Airline Data -- filtered)	reduceByKey + filtered sortBy at <ipython-input-41-4f121cf53178>:9	2015/08/13 18:31:47	99 ms	1/1 (1 skipped)	11/11 (11 skipped)
33 (Airline Data -- filtered)	reduceByKey + filtered sortBy at <ipython-input-41-4f121cf53178>:9	2015/08/13 18:17:58	14 min	2/2	22/22