STITCH FIX

Spark SQL Performance Troubleshooting and Tuning

Sky Yin

Jun 2, 2016

Quick Start

Spark is a distributed computing platform

- Distributed =/= Scalable =/= Easy to scale
  - Redshift vs Spark
- Not for data storage (except caching)
- 4 horse chariot
  - 4 language support: Java, Scala, Python, R
  - 4 use cases:
    - SQL
    - Machine learning
    - Graph analysis
    - Stream processing

Quick Start

Two ways to run a Spark SQL query

- Flotilla
  - PySpark notebook (not ideal, better to use Presto)
- `ETL` command
  - Demo 1: https://github.com/stitchfix/quant_workshop/tree/master/sparksql
  - How does it run? `ETL` => Genie => Spark cluster

Quick Start

Resources

- Our Redshift -> Spark migration guide (thanks Doug, Daragh, Matt, Sky and etc.)
- Spark SQL function reference
  - `pyspark.sql.functions module` section in PySpark doc
- Stack Overflow #apache-spark-sql
- Slack channel #spark-users

Understanding Spark SQL Execution

Fundamental building block: RDD

- Resilient Distributed Dataset
- RDD partition =/= Hive metastore partition
- Everything is RDD
  - SQL query

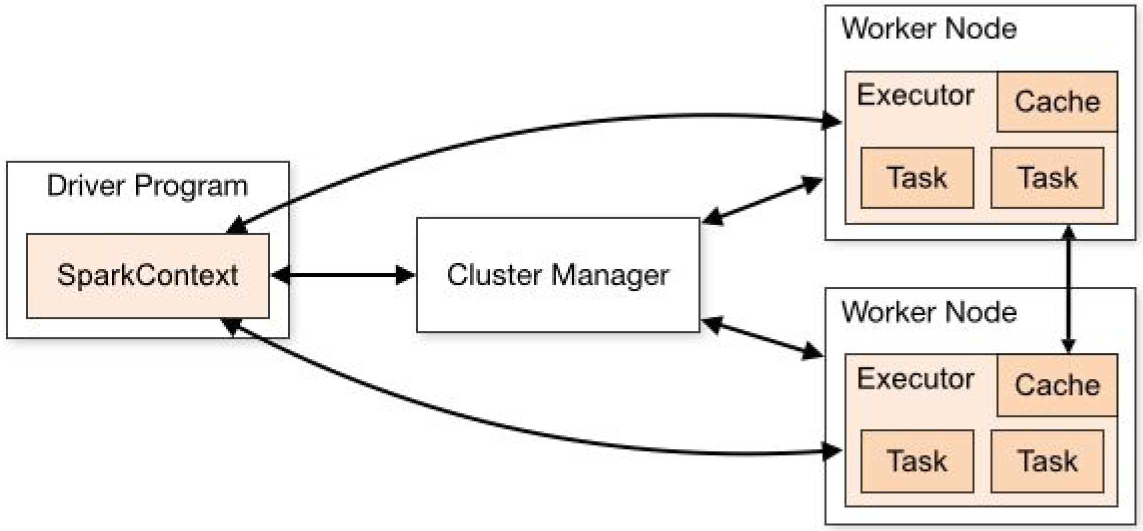
    => a graph of RDD transformation and action

    => Output

Understanding Spark SQL Execution

Key question: logical plan => physical plan?

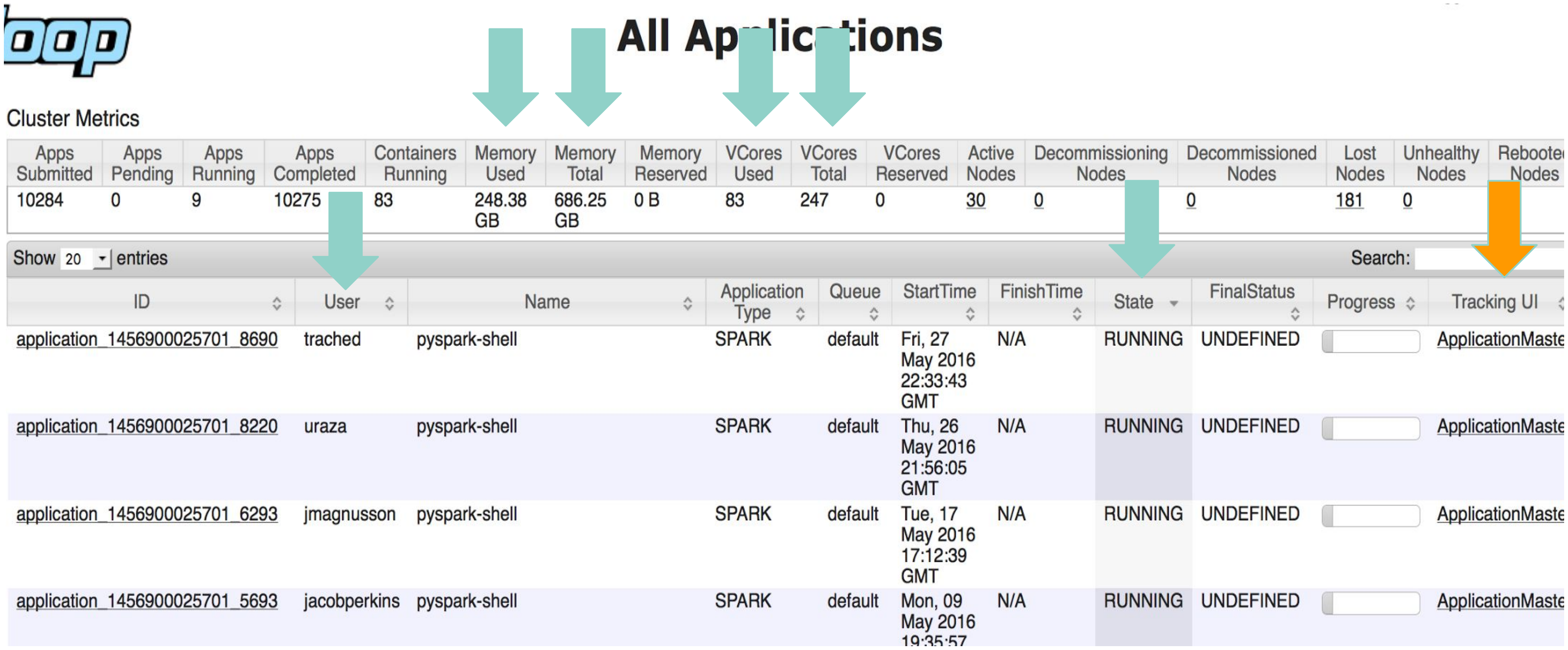
- Demo 2
- Spark execution model
- Physical plan
    - Jobs => Stages => Tasks



Monitoring Spark SQL Performance

Spark console

- `go/spark-console` (thanks Tarek)



All Applications

Cluster Metrics

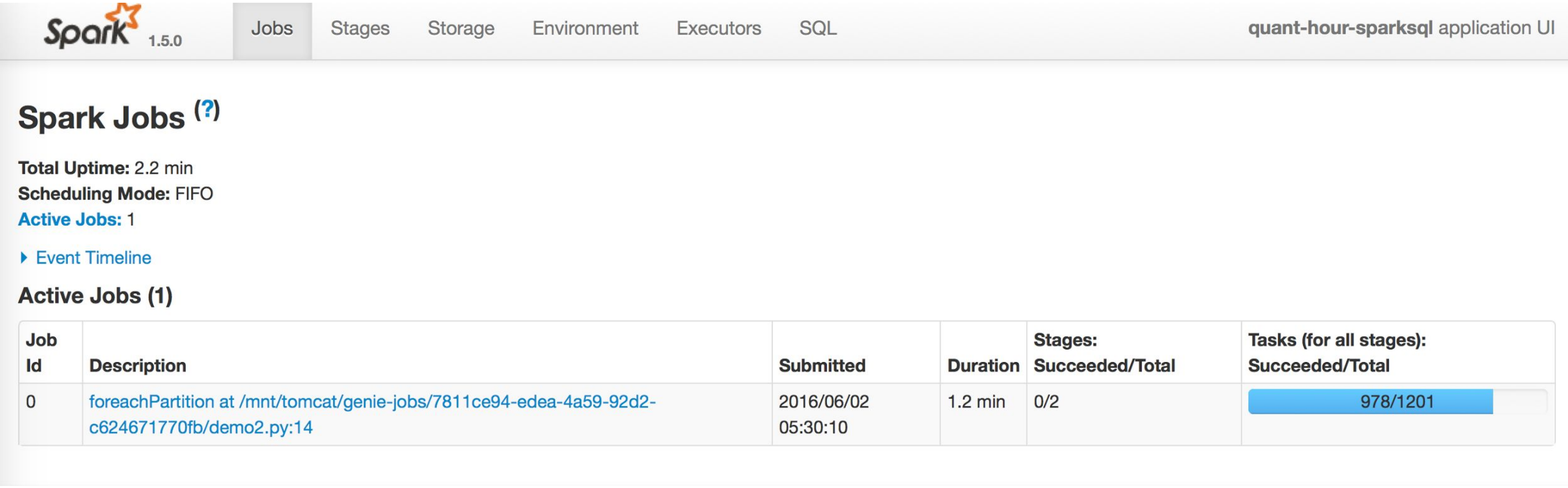
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebootec Nodes
10284	0	9	10275	83	248.38 GB	686.25 GB	0 B	83	247	0	30	0	0	181	0	

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1456900025701_8690	trached	pyspark-shell	SPARK	default	Fri, 27 May 2016 22:33:43 GMT	N/A	RUNNING	UNDEFINED		ApplicationMaste
application_1456900025701_8220	uraza	pyspark-shell	SPARK	default	Thu, 26 May 2016 21:56:05 GMT	N/A	RUNNING	UNDEFINED		ApplicationMaste
application_1456900025701_6293	jmagnusson	pyspark-shell	SPARK	default	Tue, 17 May 2016 17:12:39 GMT	N/A	RUNNING	UNDEFINED		ApplicationMaste
application_1456900025701_5693	jacobperkins	pyspark-shell	SPARK	default	Mon, 09 May 2016 19:35:57	N/A	RUNNING	UNDEFINED		ApplicationMaste

Monitoring Spark SQL Performance

Spark tracking UI



Spark Jobs (?)

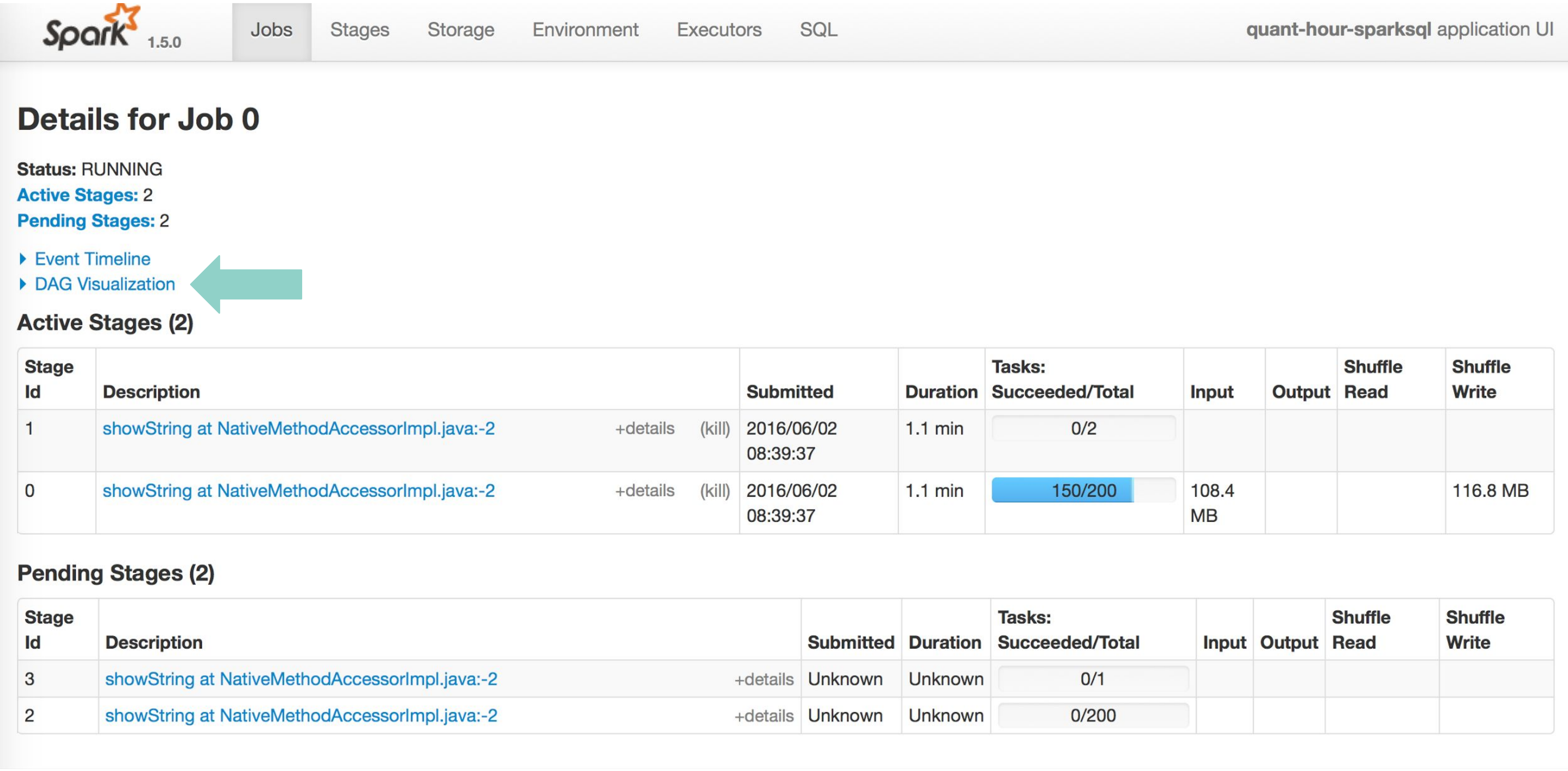
Total Uptime: 2.2 min
Scheduling Mode: FIFO
Active Jobs: 1

▸ Event Timeline

Active Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	foreachPartition at /mnt/tomcat/genie-jobs/7811ce94-edea-4a59-92d2-c624671770fb/demo2.py:14	2016/06/02 05:30:10	1.2 min	0/2	978/1201

Monitoring Spark SQL Performance



Details for Job 0

Status: RUNNING
Active Stages: 2
Pending Stages: 2

▸ Event Timeline
▸ DAG Visualization

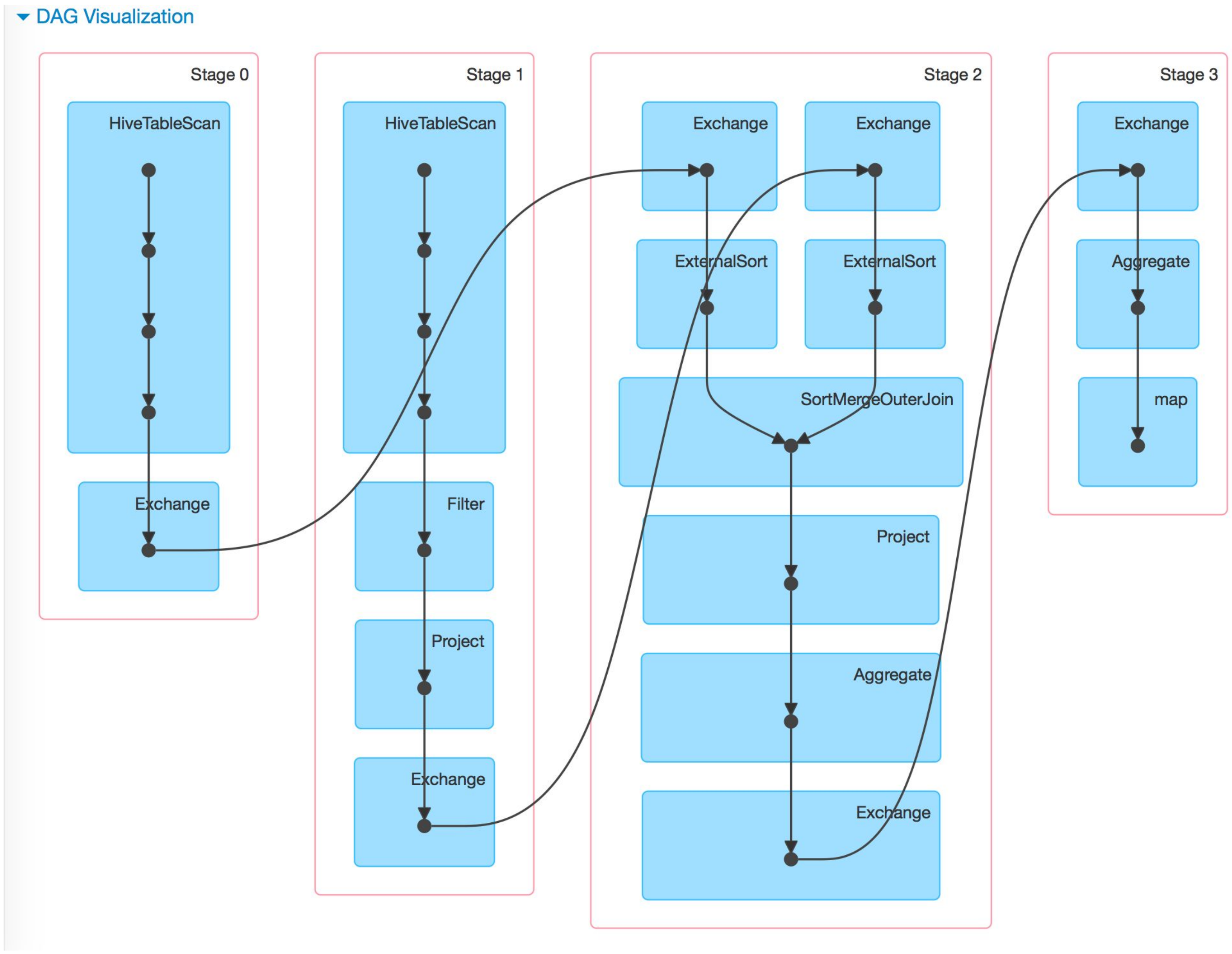
Active Stages (2)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	showString at NativeMethodAccessorImpl.java:-2 +details (kill)	2016/06/02 08:39:37	1.1 min	0/2				
0	showString at NativeMethodAccessorImpl.java:-2 +details (kill)	2016/06/02 08:39:37	1.1 min	150/200	108.4 MB			116.8 MB

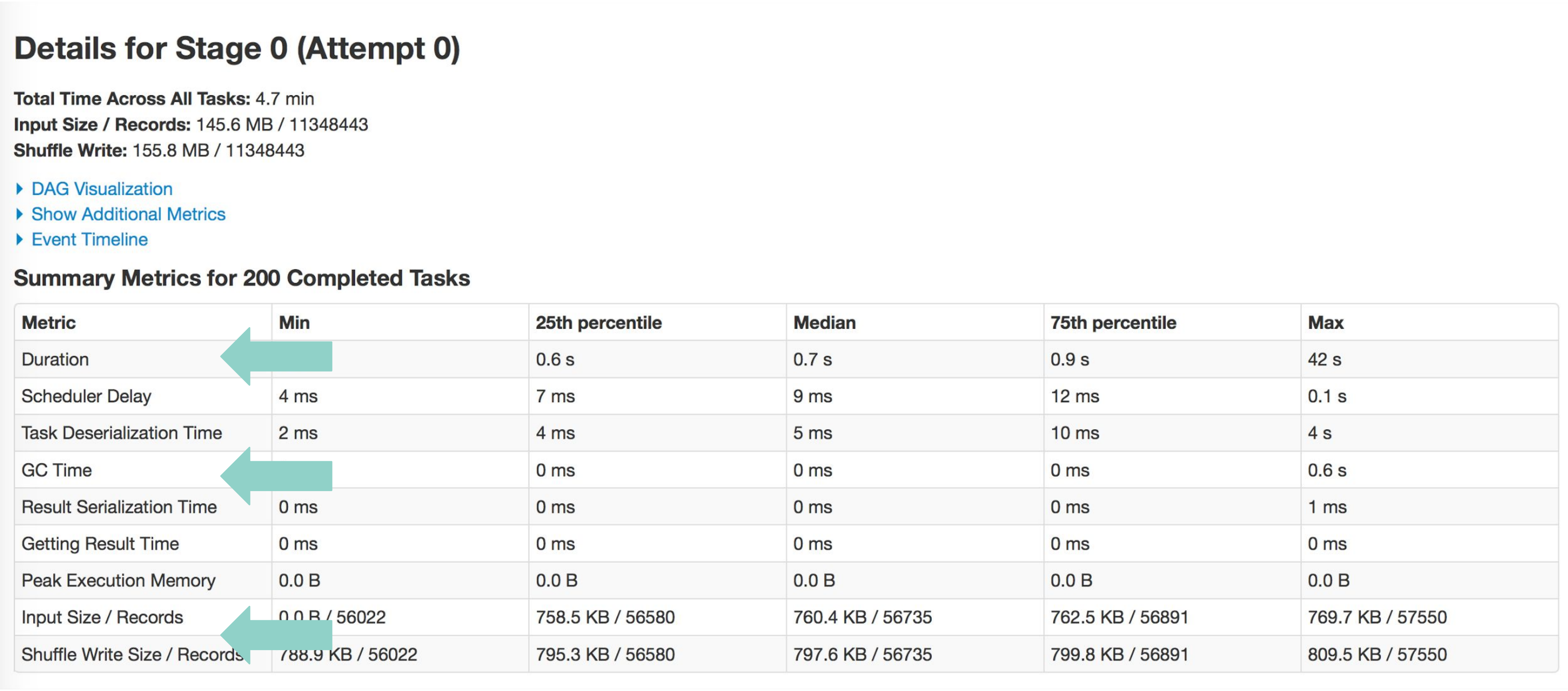
Pending Stages (2)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
3	showString at NativeMethodAccessorImpl.java:-2 +details	Unknown	Unknown	0/1				
2	showString at NativeMethodAccessorImpl.java:-2 +details	Unknown	Unknown	0/200				

Monitoring Spark SQL Performance



Monitoring Spark SQL Performance



Details for Stage 0 (Attempt 0)

Total Time Across All Tasks: 4.7 min
Input Size / Records: 145.6 MB / 11348443
Shuffle Write: 155.8 MB / 11348443

- DAG Visualization
- Show Additional Metrics
- Event Timeline

Summary Metrics for 200 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration		0.6 s	0.7 s	0.9 s	42 s
Scheduler Delay	4 ms	7 ms	9 ms	12 ms	0.1 s
Task Deserialization Time	2 ms	4 ms	5 ms	10 ms	4 s
GC Time		0 ms	0 ms	0 ms	0.6 s
Result Serialization Time	0 ms	0 ms	0 ms	0 ms	1 ms
Getting Result Time	0 ms	0 ms	0 ms	0 ms	0 ms
Peak Execution Memory	0.0 B	0.0 B	0.0 B	0.0 B	0.0 B
Input Size / Records	0.0 B / 56022	758.5 KB / 56580	760.4 KB / 56735	762.5 KB / 56891	769.7 KB / 57550
Shuffle Write Size / Records	788.9 KB / 56022	795.3 KB / 56580	797.6 KB / 56735	799.8 KB / 56891	809.5 KB / 57550

Tuning Spark SQL Performance

Parallelism

- Number of executor `--spark_num_executors`
- Number of cores/executor `--spark_executor_cores`
- Number of partition
  - Spark will run one task for each RDD partition
  - In PySpark `rdd.getNumPartitions()`
- Repartition

Tuning Spark SQL Performance

Memory

- driver vs worker
  - don’t call `collect()` on a large query result
- executor memory
  - executor loss
  - `java.lang.OutOfMemoryError: GC overhead limit exceeded`
- GC
- Caching
  - space/speed tradeoff

Tuning Spark SQL Performance

Join performance

- Try to avoid `shuffle`
  - `ShuffledHashJoin` vs `BroadcastHashJoin`
  - `spark.sql.autoBroadcastingJoinThreshold`
- `Filter` before `join`
- `Demo 3`
  - [`SortMergeJoin`]
- Detect uneven `shuffle`
  - Tasks taking much longer to run
  - Tasks with much higher input or shuffle output
- Think about the actual distribution of data
  - Ex. group by gender?