

# Structured Streaming profiling through SparkLens

[Sparklens](#) is very useful for tuning Spark Applications and also for EMR Cluster size estimations. Currently it only supports batch applications but can be extended to Spark Structured streaming.

At Intuit, we started using SparkLens for batch Jobs and working on structured streaming feature support (we are planning to contribute back to Sparklens)

As structured streaming applications will be running continuously, so we cannot directly use current `com.qubole.sparklens.QuboleJobListener`, because streaming application will be running continuously.

Each streaming Application consists of multiple batches based on Triggers and each batch might consist of `m` number of Jobs launched for each Trigger(Batch).

So, If we use current `com.qubole.sparklens.QuboleJobListener`, then it will provide job stats for huge number of jobs after terminating the streaming query, but users/developers will be interested in getting these reports continuously for every few batches.

Expectations from Structured streaming profiling

1. Keep providing these job/stage stats for every `k` batches. (Including batch wise stats)
2. Analysis of cluster usage for each batch compare to previous batch
3. InputRate vs ProcessingRate for each batch
4. If there is any possibility, get the above stats for each Job group including Job group name instead of just Job ID. So that users can identify and tune performance of high latency block of code.

For Structured streaming Spark Inbuilt `org.apache.spark.sql.streaming.StreamingQueryListener` provides below Events

1. QueryStartedEvent
2. QueryProgressEvent
3. QueryTerminatedEvent

If we observe the Structured streaming Web UI, we can see each batch can have multiple Jobs as below.

ip-10-0-0-194.vpc.internal:18080/history/application\_1523317128491\_0356/jobs/

Completed Jobs (145)

Page: 1 2 > 2 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id (Job Group) ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
144 (a5fe0835-ef83-4a0f-9128-31456f67a4f6)	id = 1e86d4c0-26b4-48a4-af0c-377ce462bd72 runId = a5fe0835-ef83-4a0f-9128-31456f67a4f6 batch = 28 <a href="#">start at &lt;console&gt;:35</a>	2018/06/14 18:33:23	24 ms	1/1	23/23
143 (a5fe0835-ef83-4a0f-9128-31456f67a4f6)	id = 1e86d4c0-26b4-48a4-af0c-377ce462bd72 runId = a5fe0835-ef83-4a0f-9128-31456f67a4f6 batch = 28 <a href="#">start at &lt;console&gt;:35</a>	2018/06/14 18:33:23	20 ms	1/1	20/20
142 (a5fe0835-ef83-4a0f-9128-31456f67a4f6)	id = 1e86d4c0-26b4-48a4-af0c-377ce462bd72 runId = a5fe0835-ef83-4a0f-9128-31456f67a4f6 batch = 28 <a href="#">start at &lt;console&gt;:35</a>	2018/06/14 18:33:23	14 ms	1/1	4/4
141 (a5fe0835-ef83-4a0f-9128-31456f67a4f6)	id = 1e86d4c0-26b4-48a4-af0c-377ce462bd72 runId = a5fe0835-ef83-4a0f-9128-31456f67a4f6 batch = 28 <a href="#">start at &lt;console&gt;:35</a>	2018/06/14 18:33:23	11 ms	1/1	1/1
140 (a5fe0835-ef83-4a0f-9128-31456f67a4f6)	id = 1e86d4c0-26b4-48a4-af0c-377ce462bd72 runId = a5fe0835-ef83-4a0f-9128-31456f67a4f6 batch = 28 <a href="#">start at &lt;console&gt;:35</a>	2018/06/14 18:33:20	3 s	2/2	248/248
139 (a5fe0835-ef83-4a0f-9128-31456f67a4f6)	id = 1e86d4c0-26b4-48a4-af0c-377ce462bd72 runId = a5fe0835-ef83-4a0f-9128-31456f67a4f6 batch = 27 <a href="#">start at &lt;console&gt;:35</a>	2018/06/14 18:33:14	21 ms	1/1	23/23
138 (a5fe0835-ef83-4a0f-9128-31456f67a4f6)	id = 1e86d4c0-26b4-48a4-af0c-377ce462bd72 runId = a5fe0835-ef83-4a0f-9128-31456f67a4f6 batch = 28 <a href="#">start at &lt;console&gt;:35</a>	2018/06/14 18:33:14	24 ms	1/1	20/20

And it's corresponding SQL Web UI is as below

## SQL

### Completed Queries

ID	Description	Submitted	Duration	Jobs
1155	<a href="#">start at &lt;console&gt;:38</a> +details	2018/06/14 20:33:22	65 ms	2886 2887 2888 2889
1154	<a href="#">start at &lt;console&gt;:38</a> +details	2018/06/14 20:33:20	2 s	2885
1153	<a href="#">start at &lt;console&gt;:38</a> +details	2018/06/14 20:33:12	67 ms	2881 2882 2883 2884
1152	<a href="#">start at &lt;console&gt;:38</a> +details	2018/06/14 20:33:10	2 s	2880
1151	<a href="#">start at &lt;console&gt;:38</a> +details	2018/06/14 20:33:02	66 ms	2876 2877 2878 2879
1150	<a href="#">start at &lt;console&gt;:38</a> +details	2018/06/14 20:33:00	3 s	2875
1149	<a href="#">start at &lt;console&gt;:38</a> +details	2018/06/14 20:32:52	72 ms	2871 2872 2873 2874
1148	<a href="#">start at &lt;console&gt;:38</a> +details	2018/06/14 20:32:50	2 s	2870
1147	<a href="#">start at &lt;console&gt;:38</a> +details	2018/06/14 20:32:43	70 ms	2866

The structured streaming sparklens proposed output looks like below.

Application ID

```
-    Batch 1
-- Job 1
--- Stage 0
--- Stage 1
:
:
--- Stage n
-- Job 2
--- Stage 0
--- Stage 1
:
:
--- Stage n

:
:

-- Job n
--- Stage 0
--- Stage 1
:
:
--- Stage n

[ All SparkLens Metrics for above Jobs ]

-- InputRate
-- ProcessRate
-- RelativeProcessingRate
-- RelativeInputRate
-- Other Stats

:
:
:

-    Batch k
-- Job 1
--- Stage 0
--- Stage 1
:
:
--- Stage n
-- Job 2
```

```
    --- Stage 0
    --- Stage 1
    :
    :
    --- Stage n

    :
    :

-- Job n
    --- Stage 0
    --- Stage 1
    :
    :
    --- Stage n

[ All SparkLens Metrics for above Jobs ]

-- InputRate
-- ProcessRate
-- RelativeProcessingRate
-- RelativeInputRate
-- Other Stats

[Cumulative metrics for K batches ]

:
:
:
```