ID2221 - Lab 2

Daniel Sharifi
dsharifi@kth.se

Antonios Mantzaris
mantza@kth.se

General Comments

Our delivery is structured within 3 separate folders, one for each task in the lab. Task 1 and 2 can be run by running “sbt run” inside the task folder. We completed task3 with a DataBricks notebook, and thus we simply saved the notebook as a HTML file to display the results.

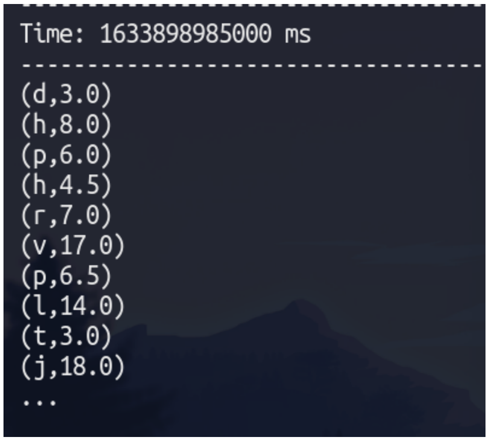
For task1 and task2 we see that the average values for all keys are converging towards 12.5, which is the expected value, considering each value is randomly sampled with equal probability in the range [1, 26].

Task 1

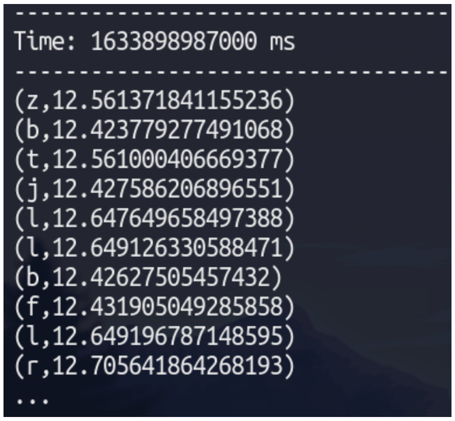
Comments:

We process the stream in task1 by processing the records in a stateful manner, with mapWithState(). To achieve this we created a case class Keeper, which keeps tracks of the cumulative sum and number of records. Each time a new record is seen in the stream the state object will be updated by increasing its count by one, and adding the new record value to the cumulative sum. Once the state is updated the mapping function returns the average by dividing the cumulative sum with the number of records.

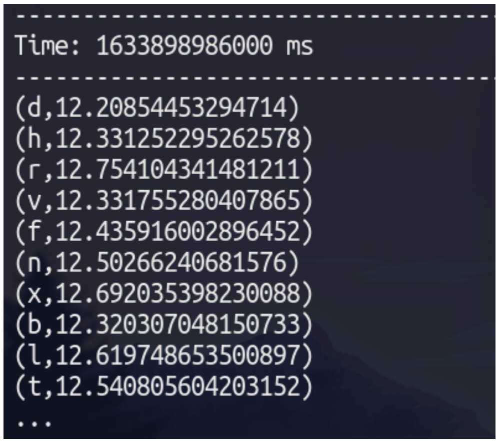
Results:



```
-----------------------------------
Time: 1633898985000 ms
-----------------------------------
(d,3.0)
(h,8.0)
(p,6.0)
(h,4.5)
(r,7.0)
(v,17.0)
(p,6.5)
(l,14.0)
(t,3.0)
(j,18.0)
...
```



```
-----------------------------------
Time: 1633898987000 ms
-----------------------------------
(z,12.561371841155236)
(b,12.423779277491068)
(t,12.561000406669377)
(j,12.427586206896551)
(l,12.647649658497388)
(l,12.649126330588471)
(b,12.42627505457432)
(f,12.431905049285858)
(l,12.649196787148595)
(r,12.705641864268193)
...
```



```
-----------------------------------
Time: 1633898986000 ms
-----------------------------------
(d,12.20854453294714)
(h,12.331252295262578)
(r,12.754104341481211)
(v,12.331755280407865)
(f,12.435916002896452)
(n,12.50266240681576)
(x,12.692035398230088)
(b,12.320307048150733)
(l,12.619748653500897)
(t,12.540805604203152)
...
```

Task2

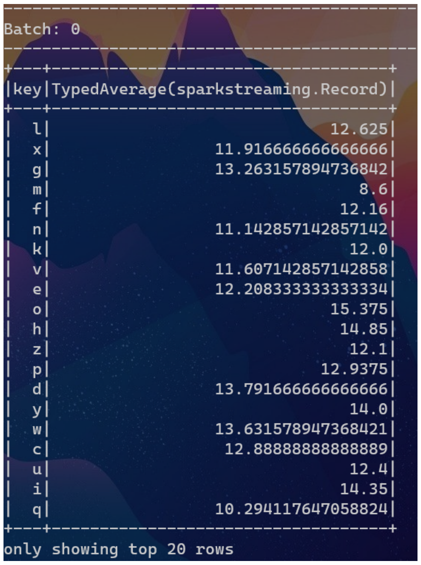
Comments:

After parsing the key value pairs into typed classes, we simply computed average by grouping records by key, and using the building aggregate tool to compute averages:

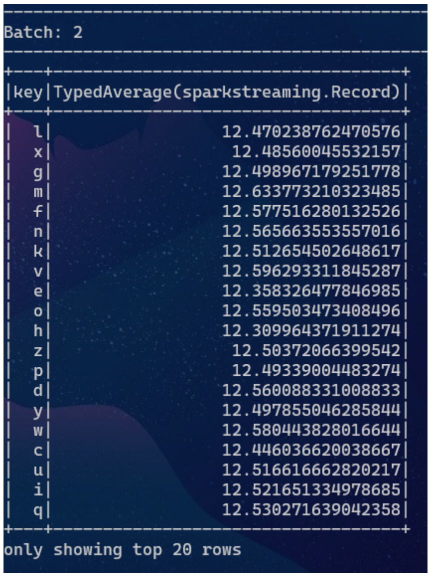
averages = ds.groupByKey(_.key).agg(typed.avg(_.value))

Aggregations such as avg are stateful by default in spark structured streaming[0].

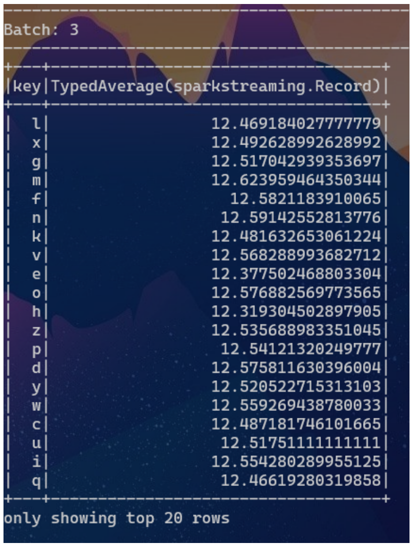
Results:



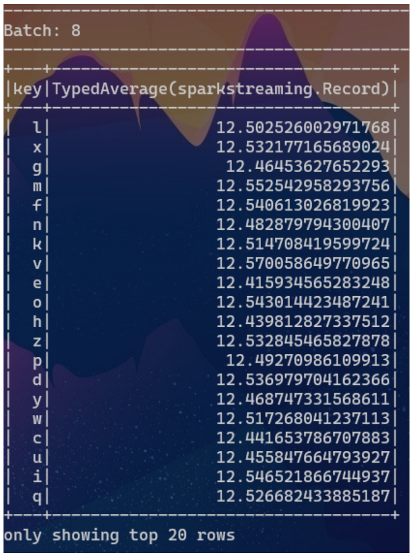
```
-------------------------------------------
Batch: 0
-------------------------------------------
+---+-----------------------------------+
|key|TypedAverage(sparkstreaming.Record)|
+---+-----------------------------------+
|  l|                             12.625|
|  x|                 11.916666666666666|
|  g|                 13.263157894736842|
|  m|                                8.6|
|  f|                              12.16|
|  n|                 11.142857142857142|
|  k|                               12.0|
|  v|                 11.607142857142858|
|  e|                 12.208333333333334|
|  o|                             15.375|
|  h|                              14.85|
|  z|                               12.1|
|  p|                            12.9375|
|  d|                 13.791666666666666|
|  y|                               14.0|
|  w|                 13.631578947368421|
|  c|                  12.88888888888889|
|  u|                               12.4|
|  i|                              14.35|
|  q|                 10.294117647058824|
+---+-----------------------------------+
only showing top 20 rows
```



```
-------------------------------------------
Batch: 2
-------------------------------------------
+---+-----------------------------------+
|key|TypedAverage(sparkstreaming.Record)|
+---+-----------------------------------+
|  l|                 12.470238762470576|
|  x|                  12.48560045532157|
|  g|                 12.498967179251778|
|  m|                 12.633773210323485|
|  f|                 12.577516280132526|
|  n|                 12.565663553557016|
|  k|                 12.512654502648617|
|  v|                 12.596293311845287|
|  e|                 12.358326477846985|
|  o|                 12.559503473408496|
|  h|                 12.309964371911274|
|  z|                  12.50372066399542|
|  p|                  12.49339004483274|
|  d|                 12.560088331008833|
|  y|                 12.497855046285844|
|  w|                 12.580443828016644|
|  c|                 12.446036620038667|
|  u|                 12.516616662820217|
|  i|                 12.521651334978685|
|  q|                 12.530271639042358|
+---+-----------------------------------+
only showing top 20 rows
```



```
-------------------------------------------
Batch: 3
-------------------------------------------
+---+-----------------------------------+
|key|TypedAverage(sparkstreaming.Record)|
+---+-----------------------------------+
|  l|                 12.469184027777779|
|  x|                 12.492628992628992|
|  g|                 12.517042939353697|
|  m|                 12.623959464350344|
|  f|                   12.5821183910065|
|  n|                  12.59142552813776|
|  k|                 12.481632653061224|
|  v|                 12.568288993682712|
|  e|                 12.377502468803304|
|  o|                 12.576882569773565|
|  h|                 12.319304502897905|
|  z|                 12.535688983351045|
|  p|                  12.54121320249777|
|  d|                 12.575811630396004|
|  y|                 12.520522715313103|
|  w|                 12.559269438780033|
|  c|                 12.487181746101665|
|  u|                  12.51751111111111|
|  i|                 12.554280289955125|
|  q|                  12.46619280319858|
+---+-----------------------------------+
only showing top 20 rows
```



```
-------------------------------------------
Batch: 8
-------------------------------------------
+---+-----------------------------------+
|key|TypedAverage(sparkstreaming.Record)|
+---+-----------------------------------+
|  l|                 12.502526002971768|
|  x|                 12.532177165689024|
|  g|                  12.46453627652293|
|  m|                 12.552542958293756|
|  f|                 12.540613026819923|
|  n|                 12.482879794300407|
|  k|                 12.514708419599724|
|  v|                 12.570058649770965|
|  e|                 12.415934565283248|
|  o|                 12.543014423487241|
|  h|                 12.439812827337512|
|  z|                 12.532845465827878|
|  p|                  12.49270986109913|
|  d|                 12.536979704162366|
|  y|                 12.468747331568611|
|  w|                 12.517268041237113|
|  c|                 12.441653786707883|
|  u|                 12.455847664793927|
|  i|                 12.546521866744937|
|  q|                 12.526682433885187|
+---+-----------------------------------+
only showing top 20 rows
```

Task 3

Open the html file attached in the task3 folder.

References:

[0] http://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#arbitrary-stateful-operations