KEYS AND GROUPS

In Spark

Big Data H/M 2022

Richard McCreadie

MOTIVATION

- So far we have discussed the application of key-less transformation functions
  - i.e. functions that work on fixed number of data items, rather than variable size groups of items that share a key

- However, as demonstrated by Google's earlier Map-Reduce proposal, there are many use-cases where you might want to have a single transformer call process a number of data items grouped by a meaningful key
  - Analyse videogames by platform
  - Group students by GPA
  - Analyse webpages by host
  - *…and many many more*

- Key-based grouping also has efficiency advantages, since all operations for a single key can be co-located on a single executer, reducing the need for data transfer

GROUPING DATA BY KEY

- As we have seen, when we load data in spark we get a Dataset<Row>, which is key-less, and we have seen previously how to use map functions to convert these datasets into other types

- To handle keys, Spark implements a more specialist Dataset type:
  - KeyValueGroupedDataset<KeyType,ValueType>

- As its name suggests, this is simply a Dataset that has been logically grouped by a set of user specified grouping keys

FROM A DATASET TO KEYVALUEGROUPEDDATASET

- So how do we transition from a Dataset<V> to KeyValueGroupedDataset<K,V>?
  - The answer is using a MapFunction<V,K>, i.e. we define a new MapFunction that extracts a key from each item in our dataset

```
map (v1) → (k1)
```

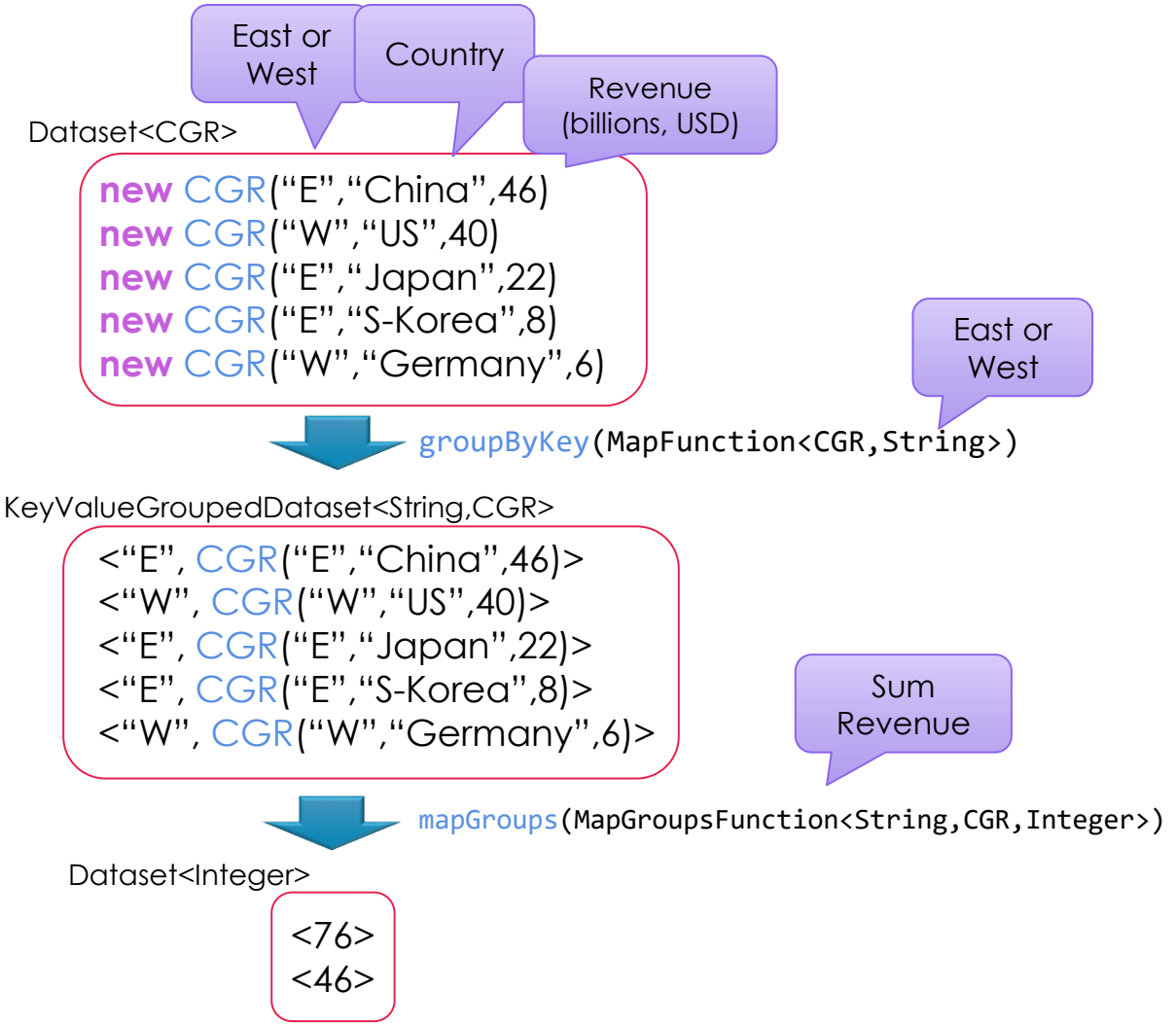
- Datasets support the groupByKey method, that uses a MapFunction like the one above to perform the conversion

```
Dataset<V>.groupByKey(MapFunction<V,K>, Encoder<K>) → KeyValueGroupedDataset<K,V>
```

MAP GROUPS

- KeyValueGroupedDatasets support the same types of operations that normal Datasets do, just based on groups
- Map groups acts as an aggregator for all items with the same key

```
mapGroups (k1,Iterator<v1>) → (v2)
```



FLATMAP GROUPS

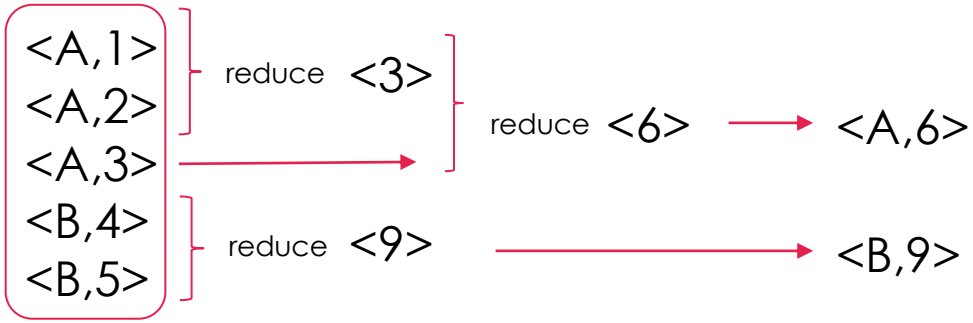
- FlatMap groups works like map groups, but returns an iterator so that 0, 1 or many items can be returned

```
mapGroups (k1,Iterator<v1>) → (v2)
flatMapGroups (k1,Iterator<v1>) → Iterator<v2>
```

- Map functions convert KeyValueGroupedDatasets back to normal datasets by combining all items that share each key

REDUCE GROUPS

- The group-based equivalent of reduce simply applies reduce for each key, returning the reduced output for each key
  - Because of this, there is **no** special ReduceGroupsFunction, a ReduceFunction is sufficient to perform reduceGroups
  - The output of a reduceGroups operation is a Dataset<Tuple2<K,V>>
    - Tuple2 is simply a Java class for holding the key value pair



SUM Example

```
KeyValueGroupedDataset.reduceGroups(ReduceFunction<V>) → Dataset<Tuple2<K,V>>
```