Quiz

1) Spark

RDD1 → union ← RDD2
↓
map → filter → save
map → filter → save

2 readings each ⓒ

2) 3 reducers

invocation of reduce?

4: Luca, Carmen, Danilo, Pablo

ⓒ

Dataset

Customers.txt

CID, Name, Surname, City, Country

TVSeries.txt

SID, Title, Genre → Can be only one

Episodes.txt

SID, SeasonNum, EpisodeNum, Title, AirD

CustomersWatched.txt

CID, StartTime, SID, SeasonNum, EpNum

→ Can watch multiple times the same episode.

MapReduce

Longest life-spanning comedy TV series.

lifespan = difference between first and last on-air date

Store SID

2 jobs (for efficiency)

job #1: # reducers > 1

mapper: emit (SID, OnAirDate)

reducer: computes min and max
compute lifespan
if lifespan > this.maxLifespan
OR (lifespan == this.max
AND SID < this.SIDMax)
{
found new max
}

in cleanup
emit (SIDMax, lifespanMax)

job #2: # reducer = 1

mapper emit (SID, lifespan)

reducer: Compute max

emit (SIDMax, NullWrit)

Spark

1) Tot # of episodes for each comedy TV series with lifespan > 3650 days.

series
↓
filter comedy
↓
map Key = SID, value = title

episode
↓
map Key = SID, value = onAirDate

→ join ←
↓
K = SID
V = title, onAir
↓
map K = SID
V = onAir, onAir, +1
↓
reduceByKey (min, max, sum)
↓
mapValues (compute lifespan)
↓
filter lifespan > 3650
↓
map K = SID
V = count Episode

2) # of TV series completely watched by each customer.

CID, # all-watched series

watched
↓
map K = CID
V = SID, Season, Episode
↓
distinct
↓
map K = CID, SID
V = +1
↓
reduceByKey sum
↓
map K = SID
V = count, CID

Series
↓
map K = SID
V = +1
↓
reduceByKey sum

→ join ←
K = SID
V = (# episodes, (# watched, CID))
↓
filter # episodes == # watched
↓
map K = CID
V = +1
↓
reduceByKey sum
↓

customers
↓
map K = CID
V = None

→ leftOuterJoin
K = CID
V = (None, count/None)
↓
mapValues
if count is None:
count = ∅
↓
save