Abbiamo cominciato scaricando pymongo per interagire con mongo tramite python.

Installato pyspark per interagire con spark.

Installato spark (con bestemmie su windows),

WINDOWS ONLY: bin hadoop (versione di spark) winutils.dll aggiunto al path (a mano), cartella spark aggiunta al path (a mano)

Creata sessione spark nel codice, aggiungendo il link al db in input e quello in output e specificando che deve scaricare e importare il modulo del connettore da spark a mongo.

Caricati i dati dal database come Dataframe spark. Dataframe è una struttura, wrapper di RDD che ha le proprietà:

1. Partizionabile
2. Località (se in memoria e tutta)

Implementato un apriori su spark che lavora partizione per partizione ed estrare gli itemset frequenti in ogni partizione con s/p supporto.

C’è da controllare effettivamente quanti elementi ci sono in ogni partizione per calcolare il supporto relativo.

Molto lento, bisogna provare a fare qualche benchmark e capire dove sia il bottleneck

Probabilmente colpa del while. A spark non sembrano piacere i cicli while

Dopo aver implementato tutto apriori su spark, siamo tornati indietro e lo abbiamo implementato sequenziale e fatto eseguire su ogni partizione. In questo modo spark non deve più fare i conti con un ciclo while e tutta la fatica per far sì che le operazioni venissero fatte partizione per partizione è sostituita da una semplice funzione (apriori)

Quando pyspark si connette a mongo, crea tante partizioni dei dati quanti processori (o shard?); quando i dati sono caricati con parallelize in memoria, vengono create tante partizioni quanti core.

Per fare lo sharding di un db su mongo bisogna eseguire questi comandi mentre connessi a mongos

Sh.enableSharding(nomedb)

Sh.shardCollection(db.collection, {key:method})

In python client.admin.command('enableSharding', db.name)

client.admin.command('shardCollection', db.name + '.' + collection.name, key={'\_id': "hashed"})