Algorytmy i metody skalowalnego przetwarzania danych - laboratoria

dr Piotr Przymus, dr Mikołaj Fejzer, dr KrzysztofRykaczewski

May 31, 2021

1 Zadanie 4: Zapytania rekurencyjne

Większość współczesnych relacyjnych baz danych wspiera zapytania rekurencyjne

Ogólna składnia zapytań rekurencyjnych

```
WITH RECURSIVE cte0 (A01, . . . , A0n) AS
(seed query UNION ALL recursive query additional clauses),
[RECURSIVE] cte1(A11, ... , A1n)
... outer query with ctei (i>=0)
```

Tego typu zapytania stosunkowo łatwo policzyć używając modelu MapReduce.

Jako inspirację do rozwiązania tego zadania można wykorzystać kod służący do obliczania domknięcia przechodniego grafu znajdującego się w przykładach Spark.

```
from __future__ import print_function

import sys
from random import Random

from pyspark.sql import SparkSession

numEdges = 200
numVertices = 100
rand = Random(42)


def generateGraph():
    edges = set()
    while len(edges) < numEdges:
        src = rand.randrange(0, numVertices)
        dst = rand.randrange(0, numVertices)
```

```
        if src != dst:
            edges.add((src, dst))
    return edges


if __name__ == "__main__":
    """
    Usage: transitive_closure [partitions]
    """
    spark = SparkSession\
        .builder\
        .appName("PythonTransitiveClosure")\
        .getOrCreate()

    partitions = int(sys.argv[1]) if len(sys.argv) > 1 else 2
    tc = spark.sparkContext.parallelize(generateGraph(), partitions).cache()

    # Linear transitive closure: each round grows paths by one edge,
    # by joining the graph's edges with the already-discovered paths.
    # e.g. join the path (y, z) from the TC with the edge (x, y) from
    # the graph to obtain the path (x, z).

    # Because join() joins on keys, the edges are stored in reversed order.
    edges = tc.map(lambda x_y: (x_y[1], x_y[0]))

    oldCount = 0
    nextCount = tc.count()
    while True:
        oldCount = nextCount
        # Perform the join, obtaining an RDD of (y, (z, x)) pairs,
        # then project the result to obtain the new (x, z) paths.
        new_edges = tc.join(edges).map(lambda __a_b: (__a_b[1][1], __a_b[1][0]))
        tc = tc.union(new_edges).distinct().cache()
        nextCount = tc.count()
        if nextCount == oldCount:
            break

    print("TC has %i edges" % tc.count())

    spark.stop()
```

1.1 Zapytania do wykonania

Dla bazy danych istniejących połączeń lotniczych:

1. Wyświetl wszystkie miasta do których można dolecieć z Toronto przy

założeniu maksymalnie K przesiadek, gdzie K jest ustalane na poziomie zapytania.

2. Wyświetl wszystkie miasta do których można dolecieć z Warszawy lub Bydgoszczy przy założeniu maksymalnie K przesiadek, gdzie K jest ustalane na poziomie zapytania.

3. Wyświetl wszystkie miasta do których można dolecieć lub dojechać z Toronto przy założeniu maksymalnie K przesiadek, gdzie K jest ustalane na poziomie zapytania.

4. Wyświetl wszystkie miasta do których można dolecieć lub dojechać z Warszawy lub Bydgoszczy bez ograniczenia maksymalnej liczby przesiadek.

1.2 Alternatywne zadanie - Przygotuj referat na jeden z tematów (15 min)

Uwaga Obowiązują zapisy:

- Domknięcie przechodnie grafu.
- Local clustering component.
- Global clustering component.
- SimRank w SPARK.
- Najkrótsza ścieżka między wierzchołkami (algorytm Dijkstry).
- Średnia długość ścieżki. • Kluster Spark na lokalnej maszynie.
- Wybrane tematy z MLlib