PFLOCK Report

Andres Calderon

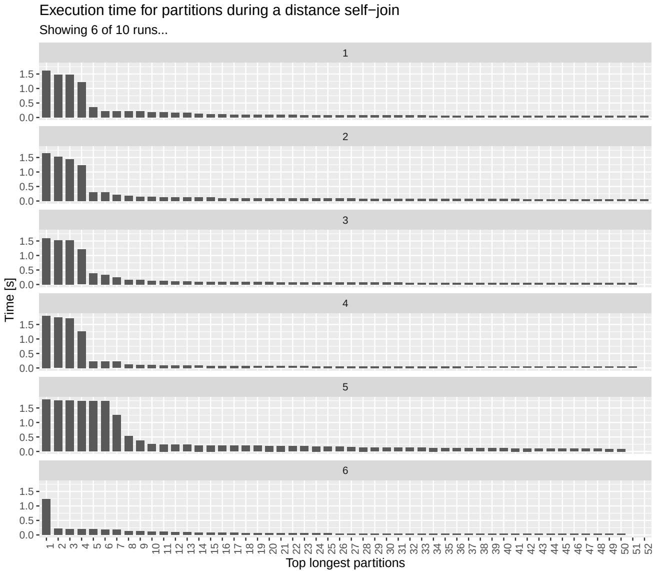
University of California, Riverside

February 7, 2020

Experiment setup...

1. Running a distance self-join to find the pairs of points far each other less than ε.
2. Using LA_25K dataset (time instant 19), quadtree partitioner and quadtree local indexing.
3. $\varepsilon = 20$, $partitions \approx 150$ (3x number of available cores).
4. 18 executors, 3 cores each.

Partitions performace...



Finding pairs algorithm...

Algorithm 1 FINDPAIRS algorithm

Require: a dataset of points $\mathcal{P}$, a number of partitions p and a distance threshold ε.

```
function FINDPAIRS ( $\mathcal{P}$, $p$, $\varepsilon$)
    Partition $\mathcal{P}$ using a Quadtree and $p$ partitions    ▷ Using Algorithm 1 in next page.
    Create a circle of radius $\varepsilon$ for each point in $\mathcal{P}$ and store them in $\mathcal{Q}$    ▷ keep same id
    Partition $\mathcal{Q}$ using the same partitioner of $\mathcal{P}$
    Build local index in $\mathcal{P}$    ▷ Using operations provided by GeoSpark
    Execute a distance join query in $\mathcal{P}$ and $\mathcal{Q}$ using $\varepsilon$ as distance    ▷ Using Algorithm 4 in following pages.
    Filter those pairs where $p_1.id < p_2.id$
end function
```

GeoSpark Partitioning algorithm...

Algorithm 1 SRDD spatial partitioning

```
Data: An original SRDD
Result: A repartitioned SRDD
/* Step 1: Build a global grid file at master node */
Take samples from the original SRDD $A$ partitions in parallel;
Construct the selected spatial structure on the collected sample at master node;
Retrieve the grids from built spatial structures;
/* Step 2: Assign grid ID to each object in parallel */
foreach spatial object in SRDD $A$ do
    foreach grid do
        if the grid intersects the object then
            Add (grid ID, object) pair into SRDD $B$;
    // Only needed for R-Tree partitioning
    if no grid intersects the object then
        Add (overflow grid ID, object) pair into SRDD $B$;
/* Step 3: Repartition SRDD across the cluster */
Partition SRDD $B$ by ID and get SRDD $C$;
Cache the new SRDD $C$ in memory and return it;
```

GeoSpark GSJoin algorithm...

Algorithm 4 *GSJoin* algorithm for range join and distance join query

```
Data: (repartitioned) SRDD A and (repartitioned) SRDD B
Result: PairRDD in schema <Left object from A, right object from B>
/* Step1: Zip partitions */
foreach partition pair from SRDD A and B with the same grid ID i do
    Merge two partitions to a bigger partition that has two sub-partitions;
Return the intermediate SRDD C;
/* Step2: Run partition-level local join */
foreach partition P in the C do
    foreach object $O_A$ in the sub-partition from A do
        if an index exists in the sub-partition from B then
            // Filter phase
            Query the spatial index of this partition using the $O_A$'s MBR;
            // Refine phase
            Check the spatial relation using real shapes of $O_A$ and candidate objects $O_B$s;
            /* Step3: Remove duplicates */
            Report <$O_A$, $O_B$> pair only if the reference point of this pair is in P;
        else
            foreach object $O_B$ in the sub-partition from B do
                Check spatial relation between $O_A$ and $O_B$;
                /* Step3: Remove duplicates */
                Report <$O_A$, $O_B$> pair only if the reference point of this pair is in P;
Generate the result PairRDD;
```