# Hadoop++

## Q1: In Hadoop++, the Trojan index is created at which granularity? Is it file-level, block-level, or tuple-level?

local level (Block-level)

## Q2: Describe the layout of a Trojan Join index over two files R and S? That is, how the data is pre-processed, organized on HDFS, and then joined at query time.



Pre-process: We apply same partition function on Join key attribute on both of relations at data loading time. Thus, co-group pairs will have same key on the same split and hence on the same node.

Join: Since relation T and S were already co-partitioned by the building Trojan Index, we can skip the shuffle and reduce phase. Join query becomes a map-only job.

## Q3: How many indexes Hadoop++ can create for a given dataset? How many of those can be “clustered” vs. “un-clustered”?

One clustered.

Many unclustered.

## Q4: If we have a Trojan index on column X of a file that consists of 200 HDFS blocks, and we have a selection query that selects all records where column X = *100.* Assume that value 100 in column X appears only in 10 blocks of the file. Answer the following:

### a) Do we need a map-only job or map-reduce job to process the above query?

map-only

### b) How many mappers will get started in Hadoop++ as part of the above query?

200

### c) How the index will save some work

Only need to check the index(the header part), which contains the key boundaries. If header part contains the X=100, then scan the data part, else, skip.

# Simba

## Q1 A key contribution in Simba is Indexing. Given the following RDD and its partitions, show the structure of the index that Simba builds on such RDD(Complete the figure to show the structure of the index.)



## Q2 For KNN query, you are given a query point (Indicated as X) in the figure below. Assume the points in the space are partitioned as illustrated in the figure, and each partition has an ID, e.g. P1



## Write down the steps that Simba will take to compute the KNN for the given query point under two cases(indicate what search boundaries will be set and which partitions will be touched.)

#### case1 K = 2:



Step 1: In case K =2, P6 is sufficient to support 2 points. The Simba will find smallest circle that cover the P6 and return all partitions which intersect with circle (dashed line circle in plot). It returns {P1,P4,P6,P7}.

Step 2: Narrow down the search area to only reach 2 point (solid line circle), and return intersected partitions. It returns {P6}.

#### case 2 K =4:



Step 1: In case K =4, P6 and P7 are sufficient to support 4 points. The Simba will find smallest circle that cover the P6 and P7 and return all partitions which intersect with circle (dashed line circle in plot). It returns {P1,P4,P6,P7,P8}.

Step 2: Narrow down the search area only reach 4 points (solid line circle), and draw a circle that covers these 4 points then return intersected partitions. It returns {P1,P6,P7}. (P4 is also possible.)