The main target is still calculating all skyline points. We must try to reduce the number of comparisons.

If we partition by space, one node partitioned is not able to include all instance of some object.

Equation 2-> upper bound

Equation 3-> upper bound

-------- upper bound for one instance.

p-skyline is not perfect for mapreduce. Since pruning is not easily cooperated with all machines in a cluster. If one machine decide to discard one point, it is hard to tell other companions.

First maprudce : partition objects into several machines. Reduce: rectangle stabbing, compute intermediate skyline results.

The second phase is the same as in the Baseline Method.

The key issue is any optimization how to partition objects to help improve efficiency.

Parallel KD-tree, using ICDE09’s method to compute the Prob-skyline of every instance.

One object has a bounding box, which could be treated as a leaf in R-tree Index.

The first phase only considers Tree traversal, and let other reducers operate 2), 3), 4) operations. After performing computation, output the result to a HDFS file. We may not consider pruning the 4) scenario.

R tree: internal nodes overlap, but leaves do not.

R+ Tree: Internal nodes don’t overlap, but leaves do overlap.

KD-Tree’s advantage: it even split the number of instances always in 2. It will get the best performance, because it can get the parent’s information very well.

An alternative is cluster one rectangle is an object, and to deploy all objects in R-tree, to devise some heuristics for load balancing.

But the problem of range query still exists.

Why not discard MapReduce Framework ? Standard distributed architecture is better for Probabilistic Skyline.