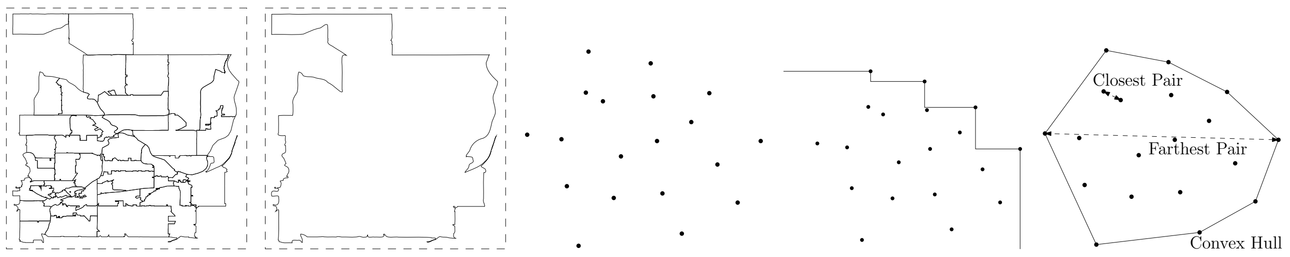

(a) Input Polygons (b) Union (c) Input Points (d) Skyline (e) Others

Figure 1: Computational Geometry Operations covered by CG_Hadoop

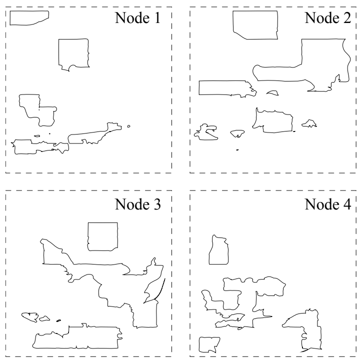

Figure 2: Polygon union in Hadoop

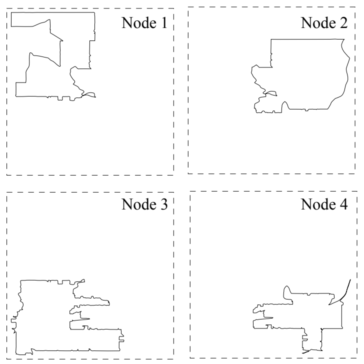

Figure 3: Polygon union in SpatialHadoop

3.1 Union in Hadoop

The main idea of our Hadoop polygon union algorithm is to allow each machine to accumulate a subset of the polygons, and then let a single machine combine the results from all machines and compute the final answer. Our algorithm works in three steps: *partitioning*, *local union*, and *global union*. The *partitioning* step distributes the input polygons into smaller subsets each handled by one machine. This step is performed by the Hadoop `load file` command which splits the file into chunks of 64MB storing each one on a slave node. In the *local union* step, each machine computes the union of its own chunk using a traditional in-memory polygon union algorithm. As each chunk is at most of size 64MB, the in-memory algorithm works fine regardless of the size of the input file. This step is implemented in Hadoop as a *combine* function, which runs locally in each machine. After performing the local union, each machine ends up with a set of polygons that represent the union of all polygons assigned to this machine. The *global union* step is implemented in Hadoop as a *reduce* function, which runs on a single machine to compute the final answer. The reduce function takes the output of all local unions, combines them into one list, and computes their union using the traditional in-memory algorithm. Each machine will end up with only few polygons, making it possible to do the union using the in-memory algorithm.

By taking advantage of a set of parallel machines, rather than performing all the work in a single machine, our proposed algorithm achieves orders of magnitude better performance than that of traditional algorithms. Although there is an overhead in partitioning the data to multiple machines, and then collecting the answer from each machine, such overhead is offset by the cost saving over parallel machines, which can be seen in large-scale spatial data sets. For interested readers, who are familiar with MapReduce programming paradigm, the pseudocode of our Hadoop polygon union algorithm is given in Appendix A.1.

Figure 2 gives the partitioning and local union steps of the input dataset of Figure 1(a) over four cluster computing nodes, where each polygon is assigned to one node. The decision of which node belongs to which partition is completely taken by the Hadoop load file component, where it basically assigns polygons to nodes randomly. As a result, and as can be seen in the figure, some polygons assigned to one node might remain completely disjoint after computing the union. In this case, all these polygons are written to the output. Then, all nodes send their output to a single machine which computes the final answer as given in Figure 1(b)

3.2 Union in SpatialHadoop

Our polygon union algorithm in SpatialHadoop has the same three steps as our algorithm in Hadoop. The only difference is that the partitioning step in SpatialHadoop is done in a spatially-aware manner, as given in Figure 3, where adjacent polygons are assigned to the same machine. The main reason here is that we utilize the underlying index structure in SpatialHadoop to be able to distribute polygons over nodes. In particular, we use the R-tree indexing in SpatialHadoop, where the size of each R-tree node is 64MB, to dump all the entries in each R-tree node to one node cluster. Since, by definition, an R-tree node provides a cluster of adjacent polygons, especially, that all R-trees in SpatialHadoop are bulk loaded, then we guarantee that all polygons in the same node are adjacent.

Although the local and global union steps remain the same, they become much lighter. The local union step mostly produces one output polygon, rather than a set of polygons as in Hadoop, therefore, the global union step processes fewer polygons. In our example, the number of polygons resulting from the local union step drops from 28 polygons in Hadoop to only four polygons in SpatialHadoop making the whole algorithm significantly faster. The pseudocode for the polygon union algorithm in SpatialHadoop is exactly the same as that of Hadoop (Appendix A.1).