Hybrid Indexing for Parallel Analysis of Spatiotemporal Point Patterns

Alexander Hohl[1], Irene Casas[2], Eric M. Delmelle[1], Wenwu Tang[1]

[1] University of North Carolina at Charlotte, Department of Geography and Earth Sciences and Center for Applied GIScience, Charlotte, NC, 28223, USA. Email: {ahohl; Eric.Delmelle; WenwuTang}@uncc.edu
[2] Louisiana Tech University, Department of Social Sciences, Ruston, LA, 71272, USA. Email: icasas@latech.edu

Abstract

High-performance parallel computing outperforms desktop workstations for computationally demanding problem solving. Domain decomposition and spatial indexing are widely used to accelerate spatial search. A single index method for spatiotemporal data processing lacks retrieval efficiency for massive computation. Combining multiple indexing methods to a hybrid spatiotemporal index holds potential for addressing this data retrieval challenge. We perform adaptive octree decomposition of the spatiotemporal domain and build local *k-d* trees to accelerate nearest neighbour search for space-time kernel density estimation (STKDE). Our parallel implementation reaches substantial speedup compared to sequential processing. The hybrid index outperforms octree decomposition alone, especially at lower-levels of parallelization. This approach facilitates finer-scale computation, enabling us to discover patterns that would be hidden otherwise.

1. Introduction

Many algorithms for analyzing spatiotemporal data are computationally intensive because they often rely on extensive nearest-neighbor (NN) search. Spatial indexing methods, including R-trees and quadtrees, have long been used to accelerate spatial queries (Lu and Ooi 1993). Further, parallel computing offers the capacity for efficient processing of intensive analyses. Efficient algorithms for range queries and k nearest neighbor (kNN) queries have been developed within parallel communication models, like Hadoop (Agarwal *et al*. 2016). Hering (2013) showed that performance of in-memory *k-d* trees is best for intermediate number of dimensions (6-13). In addition, kNN search was implemented using graphics processing units (GPU) (see Liang *et al*. 2010; Sismanis *et al*. 2012), where distance calculation and sorting are parallelized. Alternatively, the strategy of spatial domain decomposition is used for distributing the resulting subdomains to multiple computing resources for concurrent processing (Wilkinson and Allen 1999). However, including time in geographic models complicates requirements for spatial indexing, resulting in low retrieval efficiency (Gu 2011). Merging multiple indexing methods to form hybrid spatiotemporal indices was recently proposed by Azri *et al*. (2013). In this study, we perform octree-based recursive decomposition of the space-time domain for parallel computation of STKDE. Using *k-d* tree indexing within octree leaf nodes (Liu *et al*. 2008) we accelerate kNN search for STKDE.

2. Methods

2.1 Data

We used a dataset of dengue fever cases in Cali, Colombia for years 2011 and 2012. Each of the 11,168 records holds x- and y-coordinates and a timestamp (Delmelle *et al*. 2013). The rectangular envelope of the dataset spans 15,000m * 20,000m * 727days.

2.2 Space-Time Kernel Density Estimation

STKDE results in a 3D volume where each voxel (volumetric pixel) is assigned a density estimate based on the surrounding datapoints. Specifically:

$$\hat{f}(x,y,t) = \frac{1}{nh_s^2 h_t} \sum_i i(d_i < h_s, t_i < h_t) k_s \left(\frac{x-x_i}{h_s}, \frac{y-y_i}{h_s}\right) k_t \left(\frac{t-t_i}{h_t}\right) \quad (1)$$

Density $\hat{f}(x,y,t)$ of voxel s is estimated based on neighboring data points (x_i, y_i, t_i), which are weighted using the spatial and temporal Epanechnikov kernel functions, k_s and k_t (Epanechnikov 1969). Spatial and temporal distances between voxel and data point are given by d_i and t_i. If they are smaller than the spatial (h_s) and temporal bandwidths (h_t) respectively, the indicator function $i(d_i < h_s; t_i < h_t)$ equals 1, otherwise 0. We chose a combination of large bandwidths (h_s=2500m and h_t=14days), and a discretization level of 100m*100m*1 day, resulting in 16,442,664 voxels for NN search.

2.3 Spatiotemporal Domain Decomposition and Indexing

Accelerating NN search for STKDE, we develop a hybrid decomposition and indexing method. Figure 1 illustrates octree decomposition and *k-d* tree indexing (2D was used for illustration purpose). The black lines symbolize octree decomposition, creating subdomains of similar computational intensity (CI). The decomposition algorithm generates 8 cuboid subdomains by dividing each of the three axes into two equal parts. Decomposition continues recursively until no subdomain contains more points than the specified threshold (50 here), given the ratio between subdomain volume and combined subdomain and buffer volume stays above 0.1 (preventing unnecessarily small subdomains). Avoiding edge effects in STKDE, we implement buffers equal to h_s and h_t around all subdomains. For details, see Hohl *et al*. (2015). For each octree leaf node, we build a local *k-d* tree on the containing points (red lines in Figure 1). *K-d* tree is a binary tree structure for arranging points in k-dimensional space (Bentley 1975). It allows for efficient retrieval, and has been widely used for NN search (Azri *et al*. 2013). We use *k-d* tree index to accelerate the kNN search for each voxel for STKDE (blue crosses in Figure 1), resulting in massive queries. We quantify CI of each subdomain as a function of number of points and number of voxels within. To balance workloads, we distribute subdomains by equalizing cumulative CI among CPUs. Using computing time (T) and speedup (S) (Wilkinson and Allen 1999) as metrics, we compare two approaches: 1) octree decomposition only, 2) hybrid octree decomposition with *k-d* tree indexing. We compare impact of problem size between the two approaches by computing above metrics for sub-datasets (by resampling the dengue fever dataset to 1000, 2000, 3000, … , 11000 points), while fixing the number of processors at 100.

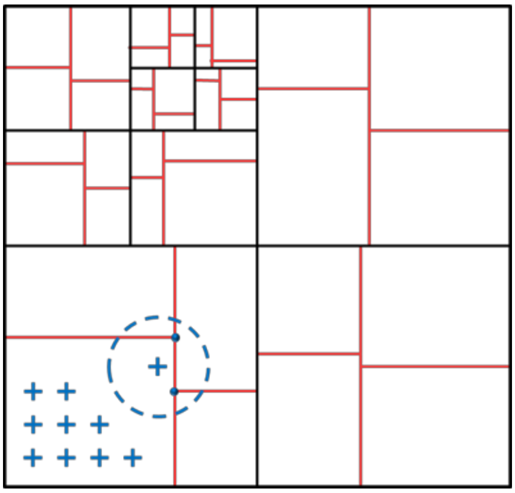

Figure 1. Spatiotemporal domain decomposition

3. Results and Discussion

Using approach 1, sequential computing time (T_s) is 40,297s. (Figure 2). When utilizing 200 CPUs in parallel, execution time (T) drops to 244s. and speedup is 165. Using approach 2, T_s is almost cut in half (22,170s.). For 200 CPUs, T is not much lower than in approach 1 (191s.), whereas speedup is lower (122). Therefore, the parallel algorithm scales better for bigger computations (approach 1) than for smaller ones (approach 2), where we used *k-d* tree indexing to reduce complexity of NN search.

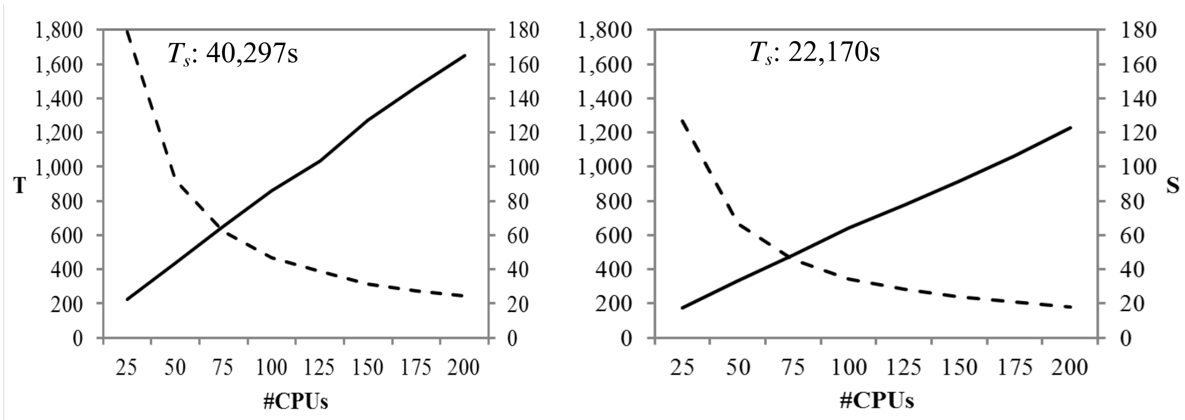

Figure 2. Performance comparison using octree decomposition only (left) and hybrid indexing (right) for 25-200 CPUs. (Speedup S: ─, Execution time *T*: ---). Sequential time: T_s.

The impact of problem size confirms above findings (Figure 3): for approach 1, T_s increases from 7,795s. (using 1,000 points) to 40,060s. (11,000 points). Utilizing 100 CPUs, T linearly increases from 210 to 457s. and the increase in speedup flattens out at around 4,000 points. Using hybrid indexing, T_s increases from 1,620s. (1,000 points) to 21,511s. (11,000 points), T linearly increases from 58s. to 332s. while speedup flattens out at 5,000 points.

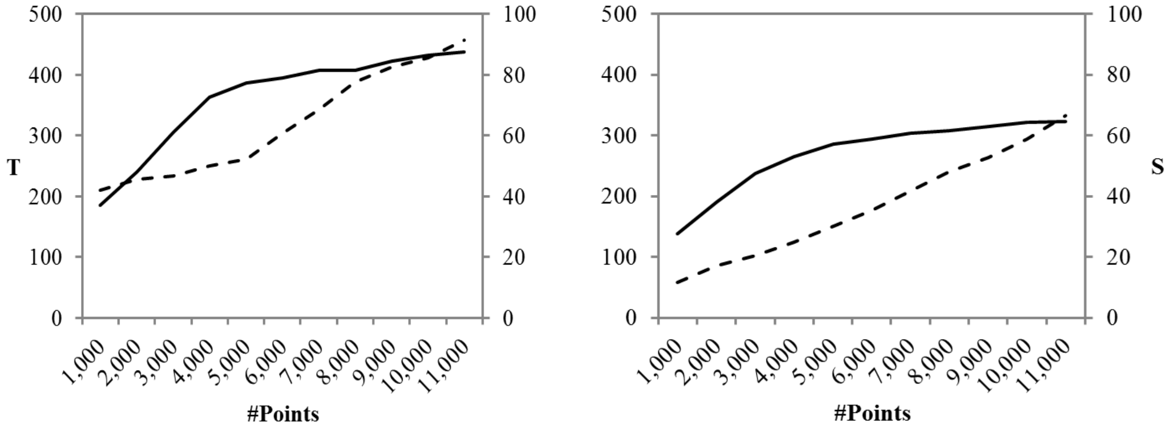

Figure 3. Performance comparison using octree decomposition only (left) and hybrid indexing (right) for 100 CPUs and 1,000 – 11,000 points. (Speedup S: ─, Execution time *T*: ---).

4. Conclusions

Our parallel implementations of STKDE reach significant speedup. Both approaches dramatically reduce computational effort for analyzing space-time patterns. As the octree decomposition only approach (approach 1) performed similarly to hybrid indexing (approach 2) for high number of CPUs, we conclude that larger computations harness better high-performance parallel computing power. Because STKDE necessitates a huge amount of spatiotemporal queries due to the discretization level, the computation is truly massive. Future experiments include bigger datasets and higher thresholds for octree decomposition, further exploring the utility of hybrid indexing in parallel spatial computing.

References

Agarwal, P. K., K. Fox, K. Munagala, and A. Nath. 2016. Parallel Algorithms for Constructing Range and Nearest-Neighbor Searching Data Structures. In Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, 429-440.

Azri, S., U. Ujang, F. Anton, D. Mioc, and A. A. Rahman. 2013. Review of Spatial Indexing Techniques for Large Urban Data Management. In International Symposium & Exhibition on Geoinformation (ISG).

Bentley, J. L. 1975. Multidimensional binary search trees used for associative searching. *Communications of the ACM* 18 (9):509-517.

Delmelle, E., I. Casas, J. H. Rojas, and A. Varela. 2013. Spatio-temporal patterns of Dengue Fever in Cali, Colombia. *International Journal of Applied Geospatial Research (IJAGR)* 4 (4):58-75.

Epanechnikov, V. A. 1969. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications* 14 (1):153-158.

Gu, W. W., Jishui; Shi, Hao; Liu Yongshan. 2011. Research on a Hybrid Spatial Index Structure. *Journal of Computational Information Systems* 7 (11):3972-3978.

Hering, T. 2013. Parallel Execution of kNN-Queries on in-memory KD Trees. In BTW Workshops. 257 - 266.

Hohl, A., E. Delmelle, and W. Tang. 2015. Spatiotemporal domain decomposition for massive parallel computation of space-time kernel density. *ISPRS Annals of the Photographer, Remote Sensing and Spatial Information Sciences* 2(4): 7.

Liang, S., Y. Liu, C. Wang, and L. Jian. 2010. Design and evaluation of a parallel k-nearest neighbor algorithm on CUDA-enabled GPU. In 2010 IEEE 2nd Symposium on Web Society.

Liu, H., Z. Huang, Q. Zhan, and P. Lin. 2008. A database approach to very large LiDAR data management. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Beijing, China* 37 (B1):463-468.

Lu, H., and B. C. Ooi. 1993. Spatial indexing: Past and future. *IEEE Data Eng. Bull*. 16 (3):16-21.

Sismanis, N., N. Pitsianis, and X. Sun. 2012. Parallel search of k-nearest neighbors with synchronous operations. In IEEE Conference on High Performance Extreme Computing (HPEC), 2012.

Wilkinson, B., and M. Allen. 1999. *Parallel programming*: Prentice hall New Jersey.