What Can You Achieve With Spatial Sorting?

Jagan Sankaranarayanan  Hanan Samet  Mike Lieberman

{jagan,hjs,codepoet}@cs.umd.edu.

Department of Computer Science
Center for Automation Research
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742
Emerging Computing Trends

- Need: Computationally hard/large problems and access to massive amounts of data
- Increasingly powerful, accessible, and programmable parallel computing
  - Programmable graphics processing units
    - NVIDIA’s Compute Unified Device Architecture (CUDA)
    - Leverage them for mathematically intensive database computation
Emerging Computing Trends

- Need: Computationally hard/large problems and access to massive amounts of data
- Increasingly powerful, accessible, and programmable parallel computing
  - Programmable graphics processing units
  - NVIDIA’s Compute Unified Device Architecture (CUDA)
  - Leverage them for mathematically intensive database computation
    1. Using simple primitives available in computations, e.g., sort, search, bit operations
    2. Little or no global synchronization among processors
Emerging Computing Trends

- Need: Computationally hard/large problems and access to massive amounts of data
- Increasingly powerful, accessible, and programmable parallel computing
  - Programmable graphics processing units
  - NVIDIA’s Compute Unified Device Architecture (CUDA)
  - Leverage them for mathematically intensive database computation
    1. Using simple primitives available in computations, e.g., sort, search, bit operations
    2. Little or no global synchronization among processors
- Challenge: How do we design algorithms for the GPU only using primitives such as sort, search, and bit operations?
Sorting in Space

- Need to understand the role of sorting in data structures and algorithms for multidimensional spatial datasets
- Spatial data is distinguished from other data types due to extent
  - Sorting implies the existence of an ordering
Sorting in Space

- Need to understand the role of sorting in data structures and algorithms for multidimensional spatial datasets
- Spatial data is distinguished from other data types due to extent
  - Sorting implies the existence of an ordering
  - Fine for one-dimensional data
    - Sort people by weight and find closest in weight to Bill and can also find closest in weight to Larry
Sorting in Space

- Need to understand the role of sorting in data structures and algorithms for multidimensional spatial datasets
- Spatial data is distinguished from other data types due to extent
  - Sorting implies the existence of an ordering
  - Fine for one-dimensional data
    - Sort people by weight and find closest in weight to Bill and can also find closest in weight to Larry
  - Role of sorting more nuanced for higher-dimensional data
    - Sort cities by distance from Chicago and find closest to Chicago but cannot find closest to New York unless resort
Sorting in Space

- Need to understand the role of sorting in data structures and algorithms for multidimensional spatial datasets

- Spatial data is distinguished from other data types due to extent
  - Sorting implies the existence of an ordering
  - Fine for one-dimensional data
    - Sort people by weight and find closest in weight to Bill and can also find closest in weight to Larry
  - Role of sorting more nuanced for higher-dimensional data
    - Sort cities by distance from Chicago and find closest to Chicago but cannot find closest to New York unless resort

- Multidimensional indexing should be implicit so we don’t have to resort the data for queries other than updates
Objects in multi-dimensional space are sorted based on the space they occupy (i.e., due to the position and extent of the objects)

Break up the space from which the objects are drawn into buckets
  - Similar to hashing

Two strategies:
  1. Object hierarchies (e.g., R-trees)
     - Sort the space based on whether it is occupied or unoccupied
  2. Disjoint cells (e.g., Quadtrees)
     - Sizes of cells are powers of two at fixed places

Objects are sorted with respect to the space they occupy which enables algorithms to prune the search space
Objects in multi-dimensional space are sorted based on the space they occupy (i.e., due to the position and extent of the objects).

Break up the space from which the objects are drawn into buckets
- Similar to hashing

Two strategies:
1. Object hierarchies (e.g., R-trees)
   - Sort the space based on whether it is occupied or unoccupied
2. Disjoint cells (e.g., Quadtrees)
   - Sizes of cells are powers of two at fixed places

Objects are sorted with respect to the space they occupy which enables algorithms to prune the search space.

Can reduce many multidimensional algorithms to sort and search processes.
What Can Be Achieved With Sorting?

1. Index Building
   - MX, Region, PR-Quadtrees (bottom-up, optimal)
   - Loose Quadtrees (gaming applications)
   - K-d Trees (using a variant of quicksort)
   - Bounding Volume Hierarchies (e.g., R-tree)

2. Joins
   - Distance Join and Semi-Joins
   - K-Nearest Neighbor (KNN) Join
   - Similarity Join

3. Well-Separated Pair (WSP) Decomposition

4. Approximate Voronoi Diagrams

5. Rectangle Intersection Problem

6. Spatial Range, Buffer, Image Dilation Queries
Quadtree Building on the GPU

Quadtrees are data structures for organizing multidimensional point objects, objects with extents (e.g., lines, polygons, regions) as well as raster images.

Applications: Many areas of computer science including Computer Graphics (e.g., octree), Databases, GIS, Computational Geometry, etc.

Vector point data:
- Input: Points drawn from $[0,1)^d$ at general positions

Raster datasets:
- Input: Monochromatic image $<x, y, color>$; color = $<0|1>$
Quadtree Building on the GPU

- Quadtrees are data structures for organizing multidimensional point objects, objects with extents (e.g., lines, polygons, regions) as well as raster images.

- Applications: Many areas of computer science including Computer Graphics (e.g., octree), Databases, GIS, Computational Geometry, etc.

- Vector point data:
  - Input: Points drawn from $[0,1)^d$ at general positions

- Raster datasets:
  - Input: Monochromatic image $<x, y, \text{color}>; \text{color} = <0|1>$

- Quadtree Variants:
  1. MX Quadtree
  2. Region Quadtree
  3. Bottom-up PR Quadtree
  4. Bottom-up Bucket PR Quadtree
  5. Top-down PR Quadtree
  6. Loose Quadtree
Quadtrees and Space-Filling Curves

- Quadtrees on GPU:
  - Pointer-based access structures require pointer indirections
  - Idea: Flatten or linearize the quadtree access structure
    1. Impose implicit access structure (e.g., sorted list)
    2. Ordering of objects using a space-filling curve
       - Use of Z-curve or Morton-curve
  - Traverse access structure using binary searches

![Diagram of Z-curve](image)

What Can You Achieve With Spatial Sorting? – p.7/14
Quadtrees and Space-Filling Curves

- Quadtrees on GPU:
  - Pointer-based access structures require pointer indirections
  - Idea: Flatten or linearize the quadtree access structure
    1. Impose implicit access structure (e.g., sorted list)
    2. Ordering of objects using a space-filling curve
      - Use of Z-curve or Morton-curve
  - Traverse access structure using binary searches

- Strategy:
  1. Place all the points or pixels in the right sequence in the final linear structure
     - Using a GPU-based sorting function
  2. Grow region or block around each point or pixel

\[
x = 6 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 = 22
\]
\[
y = 1 \quad 0 \quad 0 \quad 1
\]
Region quadtree in (b) from an input raster image in (a)
PR and Bucket PR quadtrees from an input point set
Similarity Join

- Given two object sets $A, B$ in $d$-dimensional space, find all pairs $(p \in A, q \in B)$ with distance $D(p, q) \leq \epsilon$
  - Applications in clustering, text mining, multimedia databases, ...

- Performed by:
  - Extracting high dimensional feature vectors from initial object sets
  - Measuring distance between feature vectors and returning those that satisfy distance measure
Similarity Join

- Given two object sets \( A, B \) in \( d \)-dimensional space, find all pairs \((p \in A, q \in B)\) with distance \( D(p, q) \leq \epsilon \)
  - Applications in clustering, text mining, multimedia databases, ...

- Performed by:
  - Extracting high dimensional feature vectors from initial object sets
  - Measuring distance between feature vectors and returning those that satisfy distance measure
**Similarity Join**

- Given two object sets $A, B$ in $d$-dimensional space, find all pairs $(p \in A, q \in B)$ with distance $D(p, q) \leq \epsilon$
  - Applications in clustering, text mining, multimedia databases, ...

- Performed by:
  - Extracting high dimensional feature vectors from initial object sets
  - Measuring distance between feature vectors and returning those that satisfy distance measure
Similarity Join

- Given two object sets $A, B$ in $d$-dimensional space, find all pairs $(p \in A, q \in B)$ with distance $D(p, q) \leq \epsilon$
  - Applications in clustering, text mining, multimedia databases, ...

- Performed by:
  - Extracting high dimensional feature vectors from initial object sets
  - Measuring distance between feature vectors and returning those that satisfy distance measure
**Similarity Join**

- Given two object sets $A, B$ in $d$-dimensional space, find all pairs $(p \in A, q \in B)$ with distance $D(p, q) \leq \epsilon$
  - Applications in clustering, text mining, multimedia databases, ...

- Performed by:
  - Extracting high dimensional feature vectors from initial object sets
  - Measuring distance between feature vectors and returning those that satisfy distance measure
Similarity Join Along Z-curve

1. Start with query point $q$
1. Start with query point $q$
Similitude Join Along Z-curve

1. Start with query point \( q \)
2. Order all points of \( A \) using z-curve
   - Points ordered as a, b, c, ...
Similarity Join Along Z-curve

1. Start with query point \( q \)
2. Order all points of \( A \) using z-curve
   - Points ordered as \( a, b, c, \ldots \)
3. Make \( \varepsilon \)-sized query region \( R \) around \( q \)
Similarity Join Along Z-curve

1. Start with query point $q$
2. Order all points of $A$ using z-curve
   - Points ordered as a, b, c, ...
3. Make $\epsilon$-sized query region $R$ around $q$
4. Map antipodal points of $R$ to z-curve
   - Antipodal points $\alpha, \beta$ are points of $R$ closest and farthest from origin, respectively
**Similarity Join Along Z-curve**

1. Start with query point \( q \)
2. Order all points of \( A \) using z-curve
   - Points ordered as a, b, c, \ldots
3. Make \( \epsilon \)-sized query region \( R \) around \( q \)
4. Map antipodal points of \( R \) to z-curve
   - Antipodal points \( \alpha, \beta \) are points of \( R \) closest and farthest from origin, respectively
5. Search interval of z-curve falling between mapped antipodal points and report points within distance \( \epsilon \)
   - For z-curve, interval between antipodal points guaranteed to contain all points in correct result
6. With optimizations, observed \( \approx 10x \) speedup over CPU-based methods on datasets of up to 4M points and 1024 dimensions
Well-Separated Pair Decomposition (WSPD)

- Sets of points $X, Y$ in balls of radius $r$ are $s$-well-separated iff the distance between balls is greater than $s \cdot r$

- Any set of points $S$ can be decomposed into $O(s^d n)$ $s$-well-separated pairs of subsets of $S$ (Callahan and Kosaraju 1995)
  - A pair of points is contained in exactly one well-separated pair
  - A compact ($O(n)$) way to represent $O(n^2)$ pairs of points
  - Decomposition not necessarily unique
Well-Separated Pair Decomposition (WSPD)

- Sets of points $X, Y$ in balls of radius $r$ are $s$-well-separated iff the distance between balls is greater than $s \cdot r$.

- Any set of points $S$ can be decomposed into $O(s^d n)$ $s$-well-separated pairs of subsets of $S$ (Callahan and Kosaraju 1995).
  - A pair of points is contained in exactly one well-separated pair.
  - A compact ($O(n)$) way to represent $O(n^2)$ pairs of points.
  - Decomposition not necessarily unique.
Sets of points $X, Y$ in balls of radius $r$ are $s$-well-separated iff the distance between balls is greater than $s \cdot r$.

Any set of points $S$ can be decomposed into $\mathcal{O}(s^d n)$ $s$-well-separated pairs of subsets of $S$ (Callahan and Kosaraju 1995).

- A pair of points is contained in exactly one well-separated pair.
- A compact ($\mathcal{O}(n)$) way to represent $\mathcal{O}(n^2)$ pairs of points.
- Decomposition not necessarily unique.
WSPD Construction on the GPU

- GPU-based algorithm:
  1. Choose deepest resolution $L_m$ based on data distribution and hardware capability
  2. Map $S$ to the z-curve at resolution $L_m$, forming set of quadtree blocks $B_{L_m}$, using parallel sort
  3. For $i \in \{L_m \ldots 2\}$:
     (a) For each block $x \in B_i$ (with side length $l$), in parallel:
        i. Using binary search, find all blocks $y \in B_i$ such that
        \[
        \frac{1}{2} sl \sqrt{d} \leq D(x, y) < sl \sqrt{d}
        \]
        ii. For each such block $y$, report $(x, y)$ as an $s$-well-separated pair
     (b) Form next set of quadtree blocks $B_{i-1}$ by merging adjacent sibling quadtree blocks

- Work in progress!
  - We expect similar speedups as observed for the similarity join