The SIGSPATIAL Special

The SIGSPATIAL Special is the newsletter of the Association for Computing Machinery (ACM) Special Interest Group on Spatial Information (SIGSPATIAL).

ACM SIGSPATIAL addresses issues related to the acquisition, management, and processing of spatially-related information with a focus on algorithmic, geometric, and visual considerations. The scope includes, but is not limited to, geographic information systems.

Current Elected ACM SIGSPATIAL officers are:
- Chair, Mohamed Mokbel, University of Minnesota
- Past Chair, Walid G. Aref, Purdue University
- Vice-Chair, Shawn Newsam, University of California at Merced
- Secretary, Roger Zimmermann, National University of Singapore
- Treasurer, Egemen Tanin, University of Melbourne

Current Appointed ACM SIGSPATIAL officers are:
- Newsletter Editor, Chi-Yin Chow (Ted), City University of Hong Kong
- Webmaster, Ibrahim Sabek, University of Minnesota

For more details and membership information for ACM SIGSPATIAL as well as for accessing the newsletters please visit http://www.sigspatial.org.

The SIGSPATIAL Special serves the community by publishing short contributions such as SIGSPATIAL conferences’ highlights, calls and announcements for conferences and journals that are of interest to the community, as well as short technical notes on current topics. The newsletter has three issues every year, i.e., March, July, and November. For more detailed information regarding the newsletter or suggestions please contact the editor via email at chiychow@cityu.edu.hk.

Notice to contributing authors to The SIGSPATIAL Special: By submitting your article for distribution in this publication, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights:

- to publish in print on condition of acceptance by the editor,
- to digitize and post your article in the electronic version of this publication,
- to include the article in the ACM Digital Library,
- to allow users to copy and distribute the article for noncommercial, educational or research purposes.

However, as a contributing author, you retain copyright to your article and ACM will make every effort to refer requests for commercial use directly to you.

Notice to the readers: Opinions expressed in articles and letters are those of the author(s) and do not necessarily express the opinions of the ACM, SIGSPATIAL or the newsletter.

The SIGSPATIAL Special (ISSN 1946-7729) Volume 6, Number 3, November 2014.
# Table of Contents

Message from the Editor.............................................................................................................................................. 1
Chi-Yin Chow

Section 1: Special Issue on Big Spatial Data (Part 2)

Introduction to this Special Issue: Big Spatial Data (Part 2)...................................................................................... 2
Mohamed F. Mokbel

The Ecosystem of SpatialHadoop............................................................................................................................ 3
Ahmed Eldawy and Mohamed F. Mokbel

High Performance Spatial Queries for Spatial Big Data: from Medical Imaging to GIS............................................. 11
Fusheng Wang, Ablimit Aji, and Hoang Vo

A Vision for GPU-accelerated Parallel Computation on Geo-Spatial Datasets.............................................................. 19
Sushil K. Prasad, Michael McDermott, Satish Puri, Dhara Shah, Danial Aghajarian, Shashi Shekhar, and Xun Zhou

Large-Scale Spatial Data Processing on GPUs and GPU-Accelerated Clusters.............................................................. 27
Jianting Zhang, Simin You, and Le Gruenwald

Emerging Trends in Monitoring Landscapes and Energy Infrastructures with Big Spatial Data........................................... 35
Budhendra Bhaduri, Dilip Patlolla, Ranga R. Vatsavai, Anil Cheriyadat, Wei Lu, and Rajasekar Karthik

Section 2: Event Reports

Highlights from ACM SIGSPATIAL GIS 2014 .................................................................................................................. 46
Michael Gertz, John Krumm, and Jagan Sankaranarayanan

ACM SIGSPATIAL MobiGIS 2014 Workshop Report.................................................................................................. 48
Chi-Yin Chow and Shashi Shekhar

ACM SIGSPATIAL GeoPrivacy 2014 Workshop Report............................................................................................... 49
Grant McKenzie, Carsten Keßler, and Lars Kulik

ACM SIGSPATIAL HealthGIS 2014 Workshop Report............................................................................................... 50
Daniel Goldberg, Ori Gudes, and Yaron Kanza

ACM SIGSPATIAL GIR 2014 Workshop Report......................................................................................................... 52
Christopher B. Jones and Ross S. Purves
Message from the Editor

Chi-Yin Chow
Department of Computer Science, City University of Hong Kong, Hong Kong
Email: chiychow@cityu.edu.hk

Starting from this issue, in addition to a special issue of some topic of interest to the SIGSPATIAL community, we have a new event report section that features highlights from SIGSPATIAL conferences, workshops, and affiliate chapters.

The first section is Part 2 of the special issue on “Big Spatial Data”. The associate editor of this issue is Prof. Mohamed F. Mokbel who is currently the elected Chair of the ACM SIGSPATIAL and an Associate Professor in the Department of Computer Science and Engineering, University of Minnesota, MN, USA.

The second section consists of five event reports from:

1. The 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS)
2. The 3rd ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems (ACM SIGSPATIAL MobiGIS)
3. The 1st ACM SIGSPATIAL International Workshop on Privacy in Geographic Information Collection and Analysis (ACM SIGSPATIAL GeoPrivacy)
4. The 3rd ACM SIGSPATIAL International Workshop on the Use of GIS in Public Health (ACM SIGSPATIAL HealthGIS)
5. The 8th ACM SIGSPATIAL International Workshop on Geographic Information Retrieval (ACM SIGSPATIAL GIR)

I would like to sincerely thank all the newsletter authors, Prof. Mokbel, and event organizers for their generous contributions of time and effort that made this issue possible. I hope that you will find the newsletters interesting and informative and that you will enjoy this issue.

You can download all Special issues from:

http://www.sigspatial.org/sigspatial-special
The SIGSPATIAL Special

Section 1: Special Issue on Big Spatial Data (Part 2)

ACM SIGSPATIAL
http://www.sigspatial.org
Introduction to this Special Issue:
Big Spatial Data (Part 2)

Mohamed F. Mokbel
Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA
Email: mokbel@cs.umn.edu

This issue is Part 2 of SIGSPATIAL Special issue on Spatial Big Data. The issue includes five interesting articles. The first two articles present two approaches for MapReduce-based systems built to support Big Spatial Data. In particular, the first article by Eldawy and Mokbel presents the ecosystem of SpatialHadoop, which includes the SpatialHadoop engine with a built-in support for spatial data, the Pigeon spatial language, and a visualization layer. The second article by Wang et al. presents the Hadoop-GIS system; a scalable and high performance spatial data warehousing system for large scale spatial queries, equipped with GPU-based geometric algorithms integrated into the MapReduce pipeline.

The next two articles focus on using GPUs as a means of accelerating large-scale spatial data processing. In particular, in the third article, Prasad et al. lay out a vision for accelerating geo-spatial computations and analytics using a combination of shared and distributed memory platforms, with GPUs and hundreds to thousands processing cores. The fourth article by Zhang et al. presents data parallel designs on GPU-accelerated clusters for spatial indexing, spatial joins, and various other spatial operations. The issue is concluded with an article by Bhaduri et al., discussing various trending applications of big data at Oak Ridge National Lab (ORNL).

This concludes the second part of the special issued on Big Spatial Data; all composed of ten interesting and thought-provoking articles. I would like to sincerely thank all the authors for their invited contributions.
The Ecosystem of SpatialHadoop

Ahmed Eldawy, Mohamed F. Mokbel
Department of Computer Science and Engineering, University of Minnesota, USA

Abstract

There is a recent outbreak in the amounts of spatial data generated by different sources, e.g., smartphones, space telescopes, and medical devices, which urged researchers to exploit the existing distributed systems to process such amounts of spatial data. However, as these systems are not designed for spatial data, they cannot fully utilize its spatial properties to achieve high performance. In this paper, we describe SpatialHadoop, a full-fledged MapReduce framework which extends Hadoop to support spatial data efficiently. SpatialHadoop consists of four main layers, namely, language, indexing, query processing, and visualization. The language layer provides a high level language with standard spatial data types and operations to make the system accessible to non-technical users. The indexing layer supports standard spatial indexes, such as grid, R-tree and R+-tree, inside Hadoop file system in order to speed up spatial operations. The query processing layer encapsulates the spatial operations supported by SpatialHadoop such as range query, k nearest neighbor, spatial join and computational geometry operations. Finally, the visualization layer allows users to produce images that describe very large datasets to make it easier to explore and understand big spatial data. SpatialHadoop is already used as a main component in several real systems such as MNTG, TAREEG, TAGHREED, and SHAHED.

1 Introduction

With the recent explosion in the amounts of spatial data, many researchers are trying to process these data efficiently using the distributed systems that run on hundreds of machines such as Hadoop and Hive. Unfortunately, most of these systems are designed for general data processing and this generality comes with the price of a sub-par performance with spatial data. Therefore, there are active research projects which try to extend these system to well support spatial data. Most notably, ESRI released a suit of GIS tools for Hadoop [15] which integrates Hadoop with their flagship ArcGIS product. Hadoop-GIS [2] extends Hive with a grid index and efficient implementation of range and self-join queries. Similarly, MD-HBase [12] extends HBase with Quad tree and K-d tree indexes for point datasets and support range and kNN queries.

In this work, we describe the recent work in SpatialHadoop [6], a full-fledged system for spatial data which extends Hadoop in its core to efficiently support spatial data. SpatialHadoop is available as an open source software at http://spatialhadoop.cs.umn.edu/ and has been already downloaded around 80,000 times. SpatialHadoop consists of four main layers, namely, language, indexing, query processing, and visualization. In the language layer, SpatialHadoop provides a high level language, termed Pigeon [5], which provides standard spatial data types and query processing for easy access to non-technical users. The indexing layer provides efficient spatial indexes, such as grid, R-tree, and R+-tree, which organize the data nicely in the distributed file system. The indexes are organized in two levels, one global index that partitions the data across machines, and multiple local indexes that organize records in each machine. The query processing layer encapsulates a set of spatial operations that ship with SpatialHadoop including basic spatial operations, join operations and computational
geometry operations. The \textit{visualization} layer allows users to explore big spatial data by generating images that provide bird’s-eye view on the data. SpatialHadoop is already used in several real systems, such as SHAHED \cite{7}, TAREEG \cite{3}, MNTG \cite{11}, and TAGHREED \cite{10}.

2 Overview of SpatialHadoop

Figure 1 gives an overview of SpatialHadoop. SpatialHadoop runs on a cluster containing one master node, that breaks a MapReduce job into smaller tasks, and multiple slave nodes that carry out these tasks. The core of SpatialHadoop consists of four main layers, namely, \textit{language}, \textit{indexing}, \textit{query processing}, and \textit{visualization}, described briefly below.

(1) The \textbf{Language} layer contains \textit{Pigeon} \cite{5}, a high level language with OGC-compliant spatial data types and functions. Pigeon is discussed in Section 3. (2) The \textbf{Indexing} layer provides standard spatial indexes, such as grid, R-tree, and R+-tree, which are used to store the data in an efficient way in the Hadoop Distributed File System (HDFS). Indexes are organized in two-layers, one global index that partitions data across nodes, and multiple local indexes to organize records inside each node. These indexes are made available to the MapReduce programs through two new components, namely, SpatialFileSplitter and SpatialRecordReader. The spatial indexing layer is described in Section 4. (3) The \textbf{Query Processing} layer encapsulates the spatial operations supported by SpatialHadoop. This includes \textit{basic operations, join operations}, and \textit{CG\_Hadoop} \cite{4} which is a suite of fundamental computational geometry operations. Developers and researchers can enrich this layer by implementing more advanced spatial operations. The supported operations are discussed in Section 5. (4) The \textbf{Visualization} layer provides efficient algorithms to visualize big spatial data by generating images that give a bird’s-eye view to the data. SpatialHadoop supports \textit{single level} images, which are generated at a fixed resolution, and multilevel images, which are generated at multiple resolutions to allow users to zoom in. The details of the visualization layer is provided in Section 6.

The core of SpatialHadoop is designed to serve as a backbone for applications that deal with large scale data processing. In Section 7, we describe SHAHED \cite{7} as a case study of a real system which uses SpatialHadoop to analyze and visualize large scale satellite data.
3 Language Layer: Pigeon

Most MapReduce-based systems require huge coding efforts, therefore, they provide easy high level languages that make them usable by non-technical users, such as, HiveQL [14] for Hive and Pig Latin [13] for Hadoop. SpatialHadoop does not provide a completely new language, instead, it provides, Pigeon [5], which extends Pig Latin language [13] by adding spatial data types, functions, and operations that conform to the Open Geospatial Consortium (OGC) standard [1]. In particular, we add the following:

1. **OGC-compliant spatial data types** including, `Point`, `LineString`, and `Polygon`. Since Pig Latin does not allow defining new data types, Pigeon overrides the `bytearray` data type to define spatial data types. Conversion between `bytearray` and `geometry`, back and forth, is done automatically on the fly which makes it transparent to end users.

2. **Basic spatial functions** which are used to extract useful information from a single shape; e.g., `Area` calculates the area of a polygonal shape.

3. **OGC-standard spatial predicates** which return a Boolean value based on a test on the input polygon(s). For example, `IsClosed` tests if a linestring is closed while `Touches` checks if two geometries touch each other.

4. **Spatial analysis functions** which perform some spatial transformations on input objects such as calculating the Centroid or Intersection. These functions are usually used to perform a series of transformations on input records to produce a final answer.

5. **Spatial aggregate functions** which take a set of spatial objects and return a single value which summarizes all input objects; e.g., the `ConvexHull` returns one polygon that represents the minimal convex polygon that contains all input objects.

In addition to the functions in Pigeon, we do the following changes to the language.

1. **KNN Keyword.** A new keyword `KNN` is added to perform a k-nearest neighbor query.

2. **FILTER.** To support a range query, we override the Pig Latin selection statement, `FILTER`, to accept a spatial predicate as an input and calls the corresponding procedure for range queries.

3. **JOIN.** To support spatial joins, we override the Pig Latin join statement `JOIN` to take two spatial files as input. The processing of the `JOIN` statement is then forwarded to the corresponding spatial join procedure.

4 Spatial Indexing

Traditional Hadoop stores data files in the Hadoop Distributed File System (HDFS) as heap files. This means that the data is partitioned into HDFS blocks, of 64 MB each, without taking the values of the records into consideration. While this is acceptable for traditional queries and applications, it results in a poor performance for spatial queries. There exist traditional spatial indexes, such as the R-tree [8], however, they are designed for the local file system and traditional **procedural** programming, hence, they are not directly applicable to Hadoop which uses HDFS and MapReduce **functional** programming. HDFS is inherently limited as files can be only written in sequential manner and, once written, cannot be modified.

To overcome the limitations of traditional spatial indexes, SpatialHadoop proposes a two-layer spatial index structure which consists of one *global* index and multiple *local* indexes. The global index partitions data into HDFS blocks and distributes them among cluster nodes, while local indexes organize records inside each block. The separation of global and local indexes lends itself to the MapReduce programming paradigm where the global index is used while preparing the MapReduce job while the local indexes are used for processing the map tasks. In addition, breaking the file into smaller partitions allows each partition to be indexed separately in memory and dumping it to a file in a sequential manner. SpatialHadoop uses this two-level design to build a grid index, R-tree and R+-tree.

Figure 2 shows an example of an R-tree index built in SpatialHadoop for a 400 GB dataset of all map objects in the world extracted from OpenStreetMap. Blue lines represent data while black rectangles represent partition
boundaries of the global index. As shown in this example, SpatialHadoop adjusts the size of each partition based on data distribution such that the total size of the contents of each partition is 64MB which ensures load balancing. Records in each partition are stored together as one HDFS block in one machine.

The index is constructed in one MapReduce job that runs in three phases. (1) The partitioning phase divides the space into $n$ rectangles, then, it partitions the data by assigning each record to overlapping rectangles. The challenge in this step is how to adjust these rectangles such that the contents of each partition is around 64 MB of data to fit in one HDFS block. To overcome this challenge, we first calculate the desired number of partitions by dividing the input file size $|S|$ by the HDFS block capacity $B$, i.e., $n = |S|/B$. Then, for the grid index, we partition the space using a uniform grid of size $\sqrt{n} \times \sqrt{n}$ assuming uniformly distributed data. For R-tree and R+-tree, we draw a random sample from the input file, and bulk load this sample into an in-memory R-tree of $n$ leaf nodes using the STR algorithm [9]. Then, the boundaries of the leaf nodes are used to partition the file assuming that the random sample is representative for data distribution. (2) In the local indexing phase, each partition is processed separately on a single machine and a local index is constructed in memory before it is dumped to disk. Since the partitioning phase adjusts the size of each partition to be of a single HDFS block, it becomes possible for each machine to completely load it into memory, build the index, and write it to disk in a sequential manner. (3) The final global indexing phase constructs a global index on the master node which indexes all HDFS blocks in the file using their MBRs as indexing key. The global index is kept in the main memory of the master node and it provides an efficient way to select file blocks in a specific range.

Once the data is stored efficiently in the file system as indexes, we need to add new components that allow MapReduce programs to use them. Without these new components, the traditional MapReduce components shipped with Hadoop will not be able to make use of these indexes and will treat them as heap files. Therefore, SpatialHadoop adds two new components, namely, SpatialFileSplitter and SpatialRecordReader. The SpatialFileSplitter takes a spatially indexed input file and a user-defined filter function and it exploits the global index in the input file to prune partitions that do not contribute to answer. The SpatialRecordReader takes a locally indexed partition returned by the filter function and exploits its local index to retrieve the records that match the user query. These two components allow developers to implement many spatial operations efficiently as shown in the next section.
5 Query Processing

The efficient indexes and the new MapReduce components introduced in the indexing layer give the core of SpatialHadoop that enables the possibility of efficient realization of many spatial operations. In this section, we show a few case studies of three categories of operations, namely, basic operations, join operations and computational geometry operations. Developers can follow similar techniques to add more operations such as kNN join or reverse nearest neighbor operations.

5.1 Basic Operations

SpatialHadoop contains a number of basic spatial operations such as range query and k-nearest neighbor query. A range query takes a set of spatial records $R$ and a query area $A$ as input, and returns the records that overlap with $A$. SpatialHadoop exploits the global index with the SpatialFileSplitter to select only the partitions that overlap the query range $A$. Then, it uses the SpatialRecordReader to process the local indexes in matching partitions and find matching records. Finally, a duplicate avoidance step filters out duplicate results caused by replication in the index. Although this algorithm is efficient as it quickly prunes non-relevant partitions, it takes considerable time for very small ranges due to the overhead imposed by Hadoop for starting any MapReduce job. Therefore, if the query range is very small, i.e., matches only a few partitions, the algorithm can be implemented on a single machine without starting a MapReduce job, which provides an interactive response [7, 10].

5.2 Join Operations

Join operations are usually more complex as they deal with more than one file. In a spatial join query, the input consists of two sets of spatial records $R$ and $S$ and a spatial join predicate $\theta$, e.g., overlaps, and the output is the set of all pairs $(r, s)$ where $r \in R$, $s \in S$, and the join predicate $\theta$ is true for $(r, s)$. SpatialHadoop proposes a MapReduce-based algorithm where the SpatialFileSplitter exploits the two global indexes to find overlapping pair of partitions as illustrated in Figure 3(a). The map function uses the SpatialRecordReader to exploit the two local indexes in each pair to find matching records. Finally, a duplicate avoidance step eliminates duplicate pairs in the answer caused by replication in the index.

5.3 CG_Hadoop

CG_Hadoop [4] is a suite of computational geometry operations for MapReduce. It supports five fundamental computational geometry operations, namely, polygon union, skyline, convex hull, farthest pair, and closest pair, all implemented as MapReduce algorithms. We show the skyline algorithm as an example while interesting readers can refer to [4] for further details.
In the skyline operation, the input is a set of points $P$ and the output is the set of non-dominated points. A point $p$ dominates a point $q$ if $p$ is greater than $q$ in all dimensions. CG_Hadoop adapts the existing divide-and-conquer skyline algorithm to Hadoop as a MapReduce program. Furthermore, CG_Hadoop utilizes the spatial index constructed using SpatialHadoop to prune partitions that are outside the query range. A partition $c_i$ is pruned if all points in this partition are dominated by at least one point in another partition $c_j$, in which case we say that $c_j$ dominates $c_i$. For example in Figure 3(b), $c_1$ is dominated by $c_5$ because the top-right corner of $c_1$ (i.e., best point) is dominated by the bottom-left corner of $c_5$ (i.e., worst point). The transitivity of the skyline domination rule implies that any point in $c_5$ dominates all points in $c_1$. In addition, the partition $c_4$ is dominated by $c_6$ because the top-right corner of $c_4$ is dominated by the top-left corner of $c_6$ which means that any point along the top edge of $c_6$ dominates all points in $c_4$. Since the boundaries of each partition are tight, there has to be at least one point along each edge.

6 Visualization

The visualization process involves creating an image that describes an input dataset. This is a natural way to explore spatial datasets as it allows users to find interesting patterns in the input which are otherwise hard to spot. Traditional visualization techniques rely on a single machine to load and process the data which makes them unable to handle big spatial data. SpatialHadoop provides a visualization layer which generates two types of images, namely, single level image and multilevel images, as described below.

6.1 Single Level Image Visualization

In single level image visualization, the input dataset is visualized as a single image of a user-specified image size ($\text{width} \times \text{height}$) in pixels. SpatialHadoop generates a single level image in three phases. (1) The partitioning phase partitions the data using either the default non-spatial Hadoop partitioner or using the spatial partitioner in SpatialHadoop depending on whether the data needs to be smoothed or not. Figure 4(a) shows an example of visualizing a road network without smoothing where intersecting road segments are overlapping each other, while Figure 4(b) shows the correct and desired image where intersecting road segments are merged (i.e., smoothed). If a smooth function is needed, we have to use a spatial partitioner to ensure that intersecting road segments are processed by the same machine and can be merged. (2) In the rasterize phase, the machines in the cluster process the partitions in parallel and generate a partial image for each partition. If the default Hadoop partitioner is used, each partial image has the same size of the final desired image because the partition contains data from all over the input space. On the other hand, if a spatial partitioner is used, each partial image would be of a small size according to the region covered by the associated partition. (3) In the merging phase, the partial images are combined together to produce the final image. If a non-spatial partitioner is used, partial images are overlaid as
they all have the size of the final image as shown in Figure 4(c). On the other hand, if a spatial partitioner is used, the merging phase stitches partial images together as shown in Figure 4(d).

6.2 Multilevel Image Visualization

The quality of a single level image is limited by its resolution which means users cannot zoom in to see more details. Therefore, SpatialHadoop also supports multilevel images which consist of small tiles produced at different zoom levels as shown in Figure 4(e). The input to this algorithm is a dataset and a range of zoom levels \([z_{\text{min}}, z_{\text{max}}]\) and the output is all image tiles in the specified range of levels. A naïve approach is to use the single level image algorithm to generate each tile independently but this approach is infeasible due to the excessive number of MapReduce jobs to run. For example, at zoom level 10, there will be more than one million images which would require running one million MapReduce jobs. Alternatively, SpatialHadoop provides a more efficient algorithm that runs in two phases only, partition and rasterize. (1) The partition phase scans all input records and replicates each record \(r\) to all overlapping tiles in the image according to the MBR of \(r\) and the MBR of each tile. This phase produces one partition per tile in the desired image. (2) The rasterize phase processes all generated partitions and generates a single image out of each partition. Since the size of each image tile is small, a single machine can generate that tile efficiently. This technique is used in [7] to produce temperature heat maps for NASA satellite data.

7 Case Study: SHAHED

The core of SpatialHadoop is used in several real applications that deal with big spatial data including MNTG [11], a web-based traffic generator; TAREEG [3], a MapReduce extractor for OpenStreetMap data; TAGHREED [10], a system for querying and visualizing twitter data, and SHAHED [7], a MapReduce system for analyzing and visualizing satellite data which is further discussed in this section. SHAHED is a tool for analyzing and exploring remote sensing data publicly available by NASA in a 500 TB archive. It provides a web interface (Figure 5(a)) where users navigate through the map and the system displays satellite data for the selected area.

SHAHED uses the indexing layer in SpatialHadoop to organize satellite data in a uniform grid index as the data is uniformly distributed. Furthermore, it builds an aggregate-quad-tree local index inside each grid cell to speed up both selection and aggregate queries. On top of the spatial index, it provides a multi-resolution temporal index which organizes data in days, months and years. For example, in the daily level, it builds a
separate spatial index for each day, while in the months level, it builds one index for each month. The goal is to provide efficient query processing for both small and large temporal ranges.

In the query processing layer, it provides both selection and aggregate spatio-temporal queries where the input is a data set, e.g., temperature, a spatial range represented as a rectangular region on the map and a temporal range provided as a date range on the calendar (see Figure 5(a)). In selection queries, all values in the chosen dataset and spatio-temporal range are either returned to the user as a file to download, or further processed to produce an image as shown below. In aggregate queries, only the minimum, maximum and average values are returned.

SHAHED also makes use of the visualization layer to visualize satellite data. The results of the selection query are visualized as a satellite heat map. For example, it is used to generate a temperate heat map for the whole world, as shown in Figure 5(b), which consists of a total of 500 Million points. If a date range is selected instead of a single date, an animating video is generated which shows the change of temperature over the selected time. SHAHED also uses the multilevel image visualization technique to precompute heatmaps for different datasets over the whole world and allow users to navigate these datasets on a web interface by overlaying the generated images over the world map and updating them as the user navigates.

References


---

1Please refer to an example at http://youtu.be/hHrO5VAaak8
High Performance Spatial Queries for Spatial Big Data: from Medical Imaging to GIS

Fusheng Wang\textsuperscript{1}, Ablimit Aji\textsuperscript{2}, Hoang Vo\textsuperscript{3}

\textsuperscript{1}Department of Biomedical Informatics, Department of Computer Science, Stony Brook University, USA
\textsuperscript{2}HP Labs, USA
\textsuperscript{3}Department of Mathematics and Computer Science, Emory University, USA

Abstract

Support of high performance queries on large volumes of spatial data has become increasingly important in many application domains, including geospatial problems in numerous disciplines, location based services, and emerging medical imaging applications. There are two major challenges for managing massive spatial data to support spatial queries: the explosion of spatial data, and the high computational complexity of spatial queries. Our goal is to develop a general framework to support high performance spatial queries and analytics for spatial big data on MapReduce and CPU-GPU hybrid platforms. In this paper, we introduce Hadoop-GIS – a scalable and high performance spatial data warehousing system for running large scale spatial queries on Hadoop. Hadoop-GIS supports multiple types of spatial queries on MapReduce through skew-aware spatial partitioning, on-demand indexing, customizable spatial query engine RESQUE, implicit parallel spatial query execution on MapReduce, and effective methods for amending query results through handling boundary objects. To accelerate compute-intensive geometric operations, GPU based geometric computation algorithms are integrated into MapReduce pipelines. Our experiments have demonstrated that Hadoop-GIS is highly efficient and scalable, and outperforms parallel spatial DBMS for compute-intensive spatial queries.

1 Introduction

The proliferation of cost effective and ubiquitous positioning technologies has enabled the capturing of spatially oriented data at an unprecedented scale and rate. Volunteered Geographic Information (VGI) such as OpenStreetMap [\textsuperscript{1}] further accelerates the generation of massive spatial information from community users. Analyzing large amounts of spatial data to derive values and guide decision making has become essential to business success and scientific discovery.

The rapid growth of spatial data has been driven by not only industrial applications, but also emerging scientific applications that are increasingly data- and compute-intensive. With the rapid improvement of data acquisition technologies, it has become more efficient to capture extremely large spatial data to support scientific research. For example, digital pathology imaging has become an emerging field in the past decade, where examination of high resolution scanned images of tissue specimens enables novel and more effective methods for disease diagnosis and therapy. Pathology image analysis offers a means of rapidly carrying out quantitative, reproducible measurements of micro-anatomical features in high-resolution images. Regions of micro-anatomic objects such as nuclei and cells are computed through image segmentation algorithms, represented with their boundaries, and image features are extracted from these objects. Exploring the results of such analysis involves complex queries such as spatial cross-matching or overlay, spatial proximity computations between objects, and
queries for global spatial pattern discovery; figure 1 shows several examples of frequent spatial queries. These queries often involve billions of spatial objects and extensive geometric computations. For example, spatial cross-matching is often used to compare and evaluate image segmentation algorithm results [14]. In particular, the spatial cross-matching/overlay problem involves identifying and comparing objects belonging to a wide range of different observations.

Figure 1: Examples of spatial query cases. (a) pathology imaging; (b) GIS applications.

A major requirement for the data-intensive spatial applications is fast query response which requires a scalable architecture that can query spatial data on a large scale. Another requirement is to support queries on a cost-effective architecture such as commodity clusters or cloud environments. With the rapid improvement of instrument resolutions, increased accuracy of data analysis methods, and the massive scale of observed data, complex spatial queries have become increasingly data- and compute-intensive. A typical whole slide pathology image contains more than 100 billion pixels, millions of objects, and 100 million derived image features. A single study may involve thousands of images analyzed with dozens of algorithms - with varying parameters - to generate many different result sets to be compared and consolidated, at the scale of tens of terabytes. In addition, the aforementioned VGI also enables fast and massive geospatial data collection. Besides the data scale challenge, most spatial queries involve geometric computations that are frequently compute-intensive. While the spatial filtering using minimum bounding boxes (MBBs) can be accelerated through spatial access methods, spatial refinements such as polygon intersection verification are highly expensive operations. For instance, spatial join queries such as spatial cross-matching or spatial overlay could require significant numbers of CPU operations to process. This is mainly due to the polynomial complexity of many geometric computation methods. Such compute-intensive geometric computation, combined with the big data challenge, poses significant challenges to efficient spatial applications. There is major demand for viable spatial big data solutions from diverse fields.

Traditional spatial database management systems (SDBMSs) have been used for managing and querying spatial data, through extended spatial capabilities on top of object-relational database management systems. These systems often have major limitations on querying spatial data at massive scale, although parallel RDBMS architectures [9] are available. Parallel SDBMSs tend to reduce the I/O bottleneck through data partitioning but are not optimized for compute-intensive operations such as geometric computations. Furthermore, parallel SDBMS architecture often lacks effective spatial partitioning mechanism to balance data and task loads across partitions. The high data loading overhead is another major bottleneck for SDBMS based solutions [10, 13, 14].

In contrast, MapReduce based computing model provides a highly scalable, reliable, elastic and cost-effective framework for storing and processing massive data on a cluster or in cloud environment. While the MapReduce model fits nicely with large scale problems through its key-based partitioning, spatial queries and analytics are intrinsically complex and difficult to fit into this model due to its multi-dimensional nature. Spatial
partitioning poses two major problems to be handled: spatial data skew problem and boundary object problem. The first could lead to load imbalance of tasks in distributed systems and thus result in long query response time, and the second could lead to incorrect query results if not handled properly. Furthermore, spatial query methods have to be adapted so that they can be mapped into partition based query processing framework while preserving the correct query semantics. Spatial queries are also intrinsically complex which often rely on effective access methods to reduce the search space and alleviate the high cost of geometric computations.

Meanwhile, hybrid systems combining CPUs and GPUs are becoming commonly available in commodity clusters, but the computational capacity of such systems is often underutilized. There is a general trend towards a simplified programming model such as MapReduce and hybrid computing architectures for processing massive data, but there is a significant research gap in developing new spatial querying and analytical methods to run on such architectures.

We have developed Hadoop-GIS [2, 3, 4, 5, 6, 12] – a spatial data warehousing system over MapReduce, to support highly scalable and efficient spatial queries and analytics on large scale data. Hadoop-GIS provides a framework to parallelize multiple types of spatial queries and convert them into MapReduce based query pipelines. Specifically, Hadoop-GIS offers data skew aware spatial data partitioning to achieve task parallelization, an indexing-driven spatial query engine to process spatial queries, implicit query parallelization through MapReduce, and boundary object handling to generate accurate results. In addition, we have developed GPU based spatial operators to accelerate heavy duty geometric computation, and integrated them into MapReduce based query pipelines.

2 Overview

The main objective of Hadoop-GIS is to provide a highly scalable, cost-effective, efficient spatial query processing system for data- and compute-intensive spatial applications, that can take advantage of MapReduce running on commodity clusters and CPU-GPU hybrid platforms. We first create new spatial data processing methods and pipelines with spatial partition level parallelism through the MapReduce programming model, and develop multi-level indexing methods to accelerate spatial data processing. We provide two critical components to enable such partition based parallelism by investigating effective and scalable spatial partitioning in MapReduce (pre-processing), and query normalization methods. To maximize execution performance, we fully exploit both thread-level and data-level parallelisms and utilize SIMD (Single Instruction Multiple Data) vector units to parallelize spatial operations to support object level (via grouping of many objects) and intra-object level parallelism (via breaking down an object into many smaller components), and integrate them into MapReduce pipelines. In MapReduce environment, we propose the following steps on running a typical spatial query, as shown in Algorithm 1. In step A, we effectively partition the input to generate tiles. In step B, we assign tile UIDs to spatial objects and store the objects in the Hadoop Distributed File System (HDFS). In step C we pre-process the query, and perform a preliminary filtering based on the global region index derived from the data partitioning in step A. In step D, we perform a tile based spatial query processing in which tiles run as independent MapReduce tasks in parallel. In step E, we process the boundary objects to remove duplicate objects and normalize the query result. In step F, we perform a post processing required for certain spatial query types. In step G, we perform aggregations and any additional operators, and output results to HDFS.

2.1 Real-time Spatial Query Engine

A fundamental component of Hadoop-GIS is its standalone spatial query engine. Porting a spatial database engine for such purpose is not feasible, due to its tight integration with RDBMS engine and complexity on setup and optimization. We developed a Real-time Spatial Query Engine (RESQUE) to support spatial query processing. RESQUE takes advantage of global tile indexes and local on-demand indexes to support efficient spatial queries. In addition, RESQUE is fully optimized, supports data compression, and incurs very low overhead on
Algorithm 1: Typical workflow of spatial query processing on MapReduce

A. Data/space partitioning;
B. Data storage of partitioned data on HDFS;
C. Pre-query processing (optional);
D. for tile in input_collection do
    Index building for objects in the tile;
    Tile based spatial querying processing;
E. Boundary object handling;
F. Post-query processing (optional);
G. Data aggregation;
H. Result storage on HDFS;

data loading. Thus, RESQUE is a highly efficient spatial query engine compared to a traditional SDBMS engine. RESQUE is compiled as a shared library which can be easily deployed in a cluster environment.

Hadoop-GIS takes advantage of spatial access methods for query processing with two approaches. At the higher level, Hadoop-GIS creates global region based spatial indexes of partitioned tiles for HDFS file split filtering. Consequently, for many spatial queries such as containment queries, the system can efficiently filter most irrelevant tiles through this global region index. The global region index is small and can be stored in HDFS and shared across cluster nodes through Hadoop distributed cache mechanism. At the tile level, RESQUE supports an indexing on demand approach by building tile based spatial indexes on the fly, mainly for query processing purpose, and storing index files in the main memory. Since the tile size is relatively small, index building on a single tile is fast and significantly improves spatial query processing performance. Our experiments show that index building consumes very small fraction of overall query processing cost, and it is negligible for compute-and data-intensive queries such as cross-matching.

2.2 MapReduce Based Parallel Query Execution

Instead of using explicit spatial query parallelization as summarized in [7], we take an implicit parallelization approach by leveraging MapReduce. This will much simplify the development and management of query jobs on clusters. As data is spatially partitioned, the tile name or UID forms the key for MapReduce, and identifying spatial objects of tiles can be performed in mapping phase. Depending on the query complexity, spatial queries can be implemented as map functions, reduce functions or combination of both. Based on the query types, different query pipelines are executed in MapReduce. As many spatial queries involve high complexity geometric computations, query parallelization through MapReduce can significantly reduce query response time.

2.3 Boundary Object Handling

In the past, two approaches were proposed to handle boundary objects in a parallel query processing scenario, namely Multiple Assignment and Multiple Matching [16]. In Multiple Assignment, the partitioning step replicates boundary crossing objects and assigns them to multiple tiles. In Multiple Matching, the partitioning step assigns a boundary crossing object to a single tile, but the object may appear in multiple tile pairs for spatial joins. While the Multiple Matching approach avoids storage overhead, a single tile may have to be read multiple times for query processing, which could incur increase in both computation and I/O. The Multiple Assignment approach is simple to implement with no modification to spatial computation algorithms and fits nicely to the MapReduce programming model. For example, spatial join on tiles with Multiple Assignment based partitioning can be corrected by eliminating duplicated object pairs from the query result set, which can be efficiently implemented as an additional MapReduce job [8, 16].
3 Spatial Data Partitioning

Geospatial data tends to be heavily skewed. For example, if OpenStreetMap is partitioned into 1000 x 1000 fixed size tiles, the number of objects contained in the most skewed tile is nearly three orders of magnitude more than the one in an average tile. Such large skewed tiles could significantly increase the response time in a parallel computing environment due to the straggling tiles. Thus effective and efficient spatial data partitioning is essential for scalable spatial queries running in MapReduce.

Spatial partitioning approaches generate boundary objects that cross multiple partitions, thus violating the partition independence. Spatial query processing algorithms get around the boundary problem by using a replicate-and-filter approach [9, 16] in which boundary objects are replicated to multiple spatial partitions, and side effects of such replication is remedied by filtering the duplicates at the end of the query processing phase. This process adds extra query processing overhead proportional to the number of boundary objects. Therefore, a good spatial partitioning approach should minimize the number of boundary objects.

We develop SATO [12], an effective and scalable partitioning framework which produces balanced regions while minimizing the number of boundary objects. The partitioning methods are designed for scalability, which can be easily parallelized for high performance. SATO stands for four main steps in the partitioning pipeline: Sample, Analyze, Tear, and Optimize. First, a small fraction of the dataset is sampled to identify overall global data distribution with potential dense regions. Next, the sampled data is analyzed to produce a coarse partition scheme in which each partition region is expected to contain roughly equal amounts of spatial objects. Then these coarse partition regions are passed to the partitioning component that tears the regions into more granular partitions satisfying the partition requirements. Finally, the generated partitions are analyzed to produce multi-level partition indexes and additional partition statistics which can be used for optimizing spatial queries.

SATO integrates multiple partitioning algorithms that can handle diverse datasets, and each of the algorithm has its own merits [12]. SATO also provides MapReduce based implementation of the spatial partitioning methods through two alternative approaches: top-down approach with region level parallelization, and bottom-up approach with object level parallelization.

4 MapReduce Based Spatial Query Processing

RESQUE provides the core query engine to support spatial queries, which enables us to develop a large scale spatial query processing framework based on MapReduce. Our approach is based on spatial data partitioning, tile based spatial query processing with MapReduce, and result normalization for tile boundary objects.

4.1 Spatial Join with MapReduce

Spatial join is among the most frequently used and costly queries in many spatial applications. Next, we discuss how to map spatial join queries into the MapReduce computing model. We first show an example spatial join query for spatial cross-matching in SQL, as shown in Figure 2. This query finds all intersecting polygon pairs between two sets of objects generated from an image by two different algorithms, and computes the overlap ratios (intersection-to-union ratios) and centroid distances of the pairs. The table markup_polygon represents the boundary as polygon, algorithm UID as algorithm_uid. The SQL syntax comes with spatial extensions such as spatial relationship operator ST_INTERSECTS, spatial object operators ST_INTERSECTION and ST_UNION, and spatial measurement functions ST_CENTROID, ST_DISTANCE, and ST_AREA.

For simplicity, we first present how to process the spatial join above with MapReduce ignoring boundary objects, then we return to discuss boundary handling. Input datasets are partitioned into tiles during the data loading phase, and each record is assigned a unique partition id. The spatial join query is implemented as a MapReduce query operator processed in following three steps: i) Map step: the input datasets are scanned for Map operator, and each mapper, after applying user defined function or filter operation, emits the records
with their partition id as the key along with a tag to indicate which dataset the records belong to. ii) shuffle step: records are sorted and shuffled to group the records having the same key (same partition id), and the intermediate results are materialized to local disks. iii) Reduce step: each reducer will be assigned to process a single partition, and a spatial join processing algorithm, such as plane-sweep join or index based join, is used to process the single partition. The join algorithm used for processing the single partition can be an in-memory or a disk based depending on the size of the partition. In addition, during the execution, Hadoop-GIS constructs an in-memory R*-Tree for each dataset in a partition, and uses those indexes to process spatial join query.

4.2 Support of other Query Types with MapReduce

Other types of spatial queries follow a similar processing pattern as shown in Algorithm 1. Spatial selection or containment is a simple query type in which objects geometrically contained in selection region are returned. For example, in a medical imaging scenario, users may be interested in the cell features which are contained in a cancerous tissue region. Since data is organized in partitions, containment queries can be processed in a filter-and-refine fashion. In the filter step, partitions disjoint from the query region are excluded from further processing. In the refinement step, the candidate objects are checked with the precise geometry test. The global region index is used to generate a selective table scan operation which only scans the file splits potentially containing the query results. The query would be translated into a map only MapReduce program as shown in [6]. Support of multi-way spatial join queries and nearest neighbor queries follow a similar pattern and are discussed in [5].

For K-nearest neighbors search, Hadoop-GIS provides two algorithms for an application scenario where the query is processed over a set of query objects and the cardinality of one set of objects is much smaller than the other. For example, a query in pathology imaging would, for each stem cell, find the nearest blood vessel, compute the variation of intensity of each biological property associated with the cell in respect to the distance, and return the density distribution of blood vessels around each cell. In this case the number of cells is significantly larger than the number of blood vessels. Both algorithms [5] use a replication strategy to parallelize nearest neighbor queries. Specifically, the larger cardinality dataset is partitioned and distributed over HDFS, and mappers replicate the smaller cardinality dataset to each node. Each reducer builds an in-memory index structure, such as Voronoi diagram or R-Tree, on the smaller dataset, and processes the query over the larger dataset utilizing the index.

4.3 Boundary Handling

In partition based spatial query processing, some spatial objects may lie on partition boundaries. As the partition size gets smaller, the percentage of boundary objects increases. In general, the fraction of boundary objects is inversely proportional to the size of the partition. Boundary objects pose the challenge that they belong logically to multiple disjoint partitions and would generate duplicate results.

Hadoop-GIS remedies the boundary problem in a simple but effective way. If a query requires to return complete query result, Hadoop-GIS generates a query plan which contains a pre-processing task and a post-
processing task. In the pre-processing task, the boundary objects are duplicated and assigned to multiple intersecting partitions (multiple assignment). When each partition is processed independently during query execution, the results are not yet correct due to the duplicates. In the post-processing step, results from multiple partitions will be normalized, e.g., to eliminate duplicate records by checking the object uids, which are internally assigned and globally unique. In the post-processing step, objects will go through a filtering process that eliminates duplicate records.

Intuitively, such approach would incur extra query processing cost due to the replication and duplicate elimination steps. However, this additional cost is very small and insignificant compared to the overall query processing time [6].

4.4 Performance

The RESQUE engine is highly efficient compared to traditional spatial engine [6, 5]. In particular, the on-demand R*-Tree construction cost is less than one percent of overall spatial join cost, and it does not incur any index maintenance overhead as we discard the index after processing the query. The geometric computation is the dominant cost in cross matching spatial joins. While this is difficult to support through I/O optimization oriented parallel spatial database systems, Hadoop-GIS is well suited for such computations and outperforms parallel spatial DBMS [6]. In particular, Hadoop-GIS achieves high scalability as the on-demand spatial query engine can be easily executed in multiple parallel MapReduce tasks on cluster nodes.

5 GPU Supported Spatial Queries

GPUs employ a SIMD architecture that executes the same instruction logic on a large number of cores simultaneously. Many spatial algorithms and geometry computations do not naturally fit into such parallelization model. Two alternative approaches are proposed for GPU based geometric operation, in particular, polygon intersection.

Monte-Carlo Based Method. This approach uses Monte-Carlo method for rasterization, which transforms the combined spatial space of two polygons into pixel based representation. After such transformation, the original vector geometric computation can now be performed on the pixel based representation. The intersection area thus can be determined by counting pixels belonging to both polygons. A common approach to check if a pixel is within a polygon is to use ray tracing [11] for point-in-polygon test. As the operation for each pixel is fully independent from each other, they can be effectively executed in parallel by GPU threads [15]. Rasterization resolution is critical for achieving best performance. A high resolution rasterization yields larger number of pixels, and consequently increases the compute intensity of the geometry computations. A low resolution rasterization could increase computation efficiency, but will lead to loss of accuracy.

PixelBox. A more adaptive approach [15] — named PixelBox — can reduce the computation intensity while ensuring the computation accuracy. Specifically, PixelBox first partitions the space into cells or boxes. For boxes containing edges of polygons, rasterization is performed as Monte-Carlo approach. In this way, group of pixels in a box could be tested together for the containment relationship with a polygon, and pixel level testing is performed only for edge crossing areas. Thus, the computational efficiency could be much improved. The experiments demonstrate two orders performance improvement for intersection operation compared to a single thread CPU algorithm.

Integration of GPU Based Geometric Computation into MapReduce. To support a more efficient execution on accelerated systems, we have been extending Hadoop-GIS for execution of spatial query operations with GPUs in distributed memory machines. The goal is to design an efficient bridge interface between the MapReduce program and the GPU program. Many small tasks sent to GPU may incur much overhead on communication and subdue the benefit of GPU. We propose a prediction model to decide the granularity of tasks for GPU invocation, by considering both data communication cost and execution cost for different types of spatial operations. Another goal is to achieve load balancing and data/operation aware task assignment in
the CPU/GPU hybrid environment. We first take a knowledge based approach to decide assignments to CPU or GPU, and then build efficient task migration between the CPU and the GPU in case of an unbalanced task assignment. Preliminary work is reported in [4].

6 Software

The high adaptability of the framework allows the system to be integrated into computer clusters or cloud computing environments such as Amazon EC2. Hadoop-GIS is available as a set of library functions, including input data transformation, data partitioning, spatial indexing and spatial query processing, and MapReduce based execution. It also includes pipelines for combined multiple query jobs. We implemented the core spatial indexing and querying methods in C++, and implemented the MapReduce programs in Java. We use the Hadoop streaming mechanism to bridge the communication between C++ libraries and Java based MapReduce programs. The pipelines are as set of scripts which can be easily customized. Hadoop-GIS is open source, and it can be downloaded from the web site [2].

Acknowledgments

This work is supported in part by NSF IIS 1350885, by NSF ACI 1443054, by NLM R01LM009239, and by NCI 1U24CA180924-01A1.

References

A Vision for GPU-accelerated Parallel Computation on Geo-Spatial Datasets

Prof. Sushil K. Prasad, PhD Students: Michael McDermott, Satish Puri, Dhara Shah and Danial Aghajarian  
Department of Computer Science, Georgia State University, USA  
Prof. Shashi Shekhar, Department of Computer Science, University of Minnesota, USA  
Dr. Xun Zhou, Department of Management Sciences, University of Iowa, USA

Abstract

We summarize the need and present our vision for accelerating geo-spatial computations and analytics using a combination of shared and distributed memory parallel platforms, with general-purpose Graphics Processing Units (GPUs) with 100s to 1000s of processing cores in a single chip forming a key architecture to parallelize over. A GPU can yield one-to-two orders of magnitude speedups and will become increasingly more affordable and energy efficient due to mass marketing for gaming. We also survey the current landscape of representative geo-spatial problems and their parallel, GPU-based solutions.

1 Introduction

Geo-spatial datasets are large and the related computations and analytics are computationally intensive. For example, it takes about 13 minutes for polygonal overlay of two GIS shapefiles USA Detailed Water Bodies and USA Block Group Boundaries containing about 600K polygons using ArcGIS 10.1 on Intel Core i5 processor, and it takes roughly 20 hours to compute the spatial join of a polyline table with 73M records representing the contiguous USA with itself on an Amazon EC2 instance [31]. Therefore, harnessing parallel processing capabilities of modern hardware platforms is essential. Parallel processing technology is not just for real-time monitoring and steering, making scientific and policy decisions within reasonable time, visualizing huge datasets, and for enabling memory-constrained computations which cannot otherwise be solved on sequential computers. This technology’s vast speed gains will also result in new capabilities for the masses that are barely imagined today, analogous to what airplanes have meant over ships and trains.

The modern computing platforms are parallel, distributed and heterogeneous, ranging from mobile devices to desktops to high performance computing (HPC) clusters and cloud data-centers. Mobile devices already harness the backend cloud servers over the Internet for daily chores such as mapping and navigation. These are expected to launch high-end computations and data analytics for future applications. For GIS and other scientific applications such as multi-agent epidemic simulations, hot spot detections, and overlay computations, scientists will increasingly employ the powerful shared-memory parallel processing capabilities of their laptops and desktops. The individual compute nodes of these devices (as of clusters) consist of multi-core CPUs containing tens of processing cores and many-core GPUs containing hundreds to thousands of cores, both with

1For readers interested in shared-memory parallel programming for geo-spatial problems, we are setting up a work-in-progress webpage at [http://www.cs.gsu.edu/~dimos/?q=GIS-Resources](http://www.cs.gsu.edu/~dimos/?q=GIS-Resources) which currently contains source code and input and output data for (i) 1-D interesting region discovery using C/CUDA and Hadoop+CUDA (ii) bilateral filtering using C/CUDA, and (iii) Polygonal overlay system using Java/Hadoop and C# over Azure Cloud.
shared-memory architectures. For larger scale computing over huge data sets, possibly from multiple online sources, these devices will work in tandem with dedicated clusters and will increasingly employ on-demand HPC cloud clusters (distributed-memory parallel processing). In fact, for small-to-medium scale computations, laptops/desktops would be cost-effective alternatives to the clusters as these are not readily accessible to most scientists not involved with HPC.

Some scientists have begun employing distributed processing over clusters using map-reduce programming such as Hadoop (e.g., Spatial-Hadoop [9]) and message-passing programming such as MPI (Message-Passing Interface - e.g. PySAL [32], hydrology [35]). However, these projects currently do not employ the shared memory capabilities of the compute nodes. A few have employed multi-core capabilities of CPUs using OpenMP/Pthread (e.g., for plane-sweep [23] [19]), but not GPUs. Not harnessing the many-core GPUs typically results in a loss of at least 200%-300% speedup (2-3 fold) compared to a multi-core CPU, which can be obtained with reasonable porting effort. More often, the loss is one to two orders of magnitude for many computations, but that does require medium-to-expert level parallelization effort. The latter class of computations are typically either embarrassingly parallel, such as independent bags of uniform tasks requiring little communication among the processing cores, or are regular computations, those which have regular communication patterns among neighboring tasks. Some are irregular computations with no well-defined communication pattern due to irregular spatial and/or temporal task or data distributions, but can yield an order of magnitude speedup with sophisticated data structures and algorithms.

Our vision is that the bulk of geo-spatial software packages will employ both distributed and shared memory parallel processing including the GPUs toward scalable solutions. How will this happen? Clearly, this will require the collective effort of the GIS community and collaboration with computer scientists. Computer scientists and their graduate students will be interested in such interdisciplinary collaborations because of an excellent potential for computer science research in algorithms, data structures, database, data mining and systems over interesting datasets and domain applications. Such an investment is even more sensible given that the GPU accelerators are here to stay, will continually improve both in performance and energy footprint, and will become more affordable due to mass usage by the gaming industry. The promising news is that a few groups are employing both distributed and shared memory parallel computing including GPUs (e.g., polygonal overlay over medical images using MPI and CUDA [37]).

In this article, we explain how you can recognize which of the three categories your favorite problems fall into by understanding the nature of similar problems and get a sense of the potential speedup and how much parallelization effort may be involved. You will also obtain pointers to some representative state-of-art projects, specially for the non-embarrassingly parallel problems. For embarrassing parallel problems, Section 2 illustrates several GIS and other problems, including hotspot detection problem for a fixed radius neighborhood. Section 3 describes regular computations, some related projects, and gives detailed description of interesting path/region discovery problem and our CUDA-based analytics solutions. For irregular computations, Section 4 reviews the literature with recent GPU parallelization efforts and provides details on our polygonal overlay MPI-GIS system.

2 Embarrassingly Parallel Problems

An embarrassingly parallel problem, is the easiest kind of parallelism as these problems exhibit little to no task dependency [11]. Most of these problems can be solved with little communication between parallel tasks and this makes them distinguishable from other problems which require exchanging intermediate results among several tasks. In other words, an embarrassingly parallel problem imposes the least amount of overhead to be parallelized and, therefore, usually does not suffer from parallel slowdown.

A celebrated example of an embarrassingly parallel problem is shared-memory matrix multiplication in which for two $n \times n$ matrices, the multiplication process is split into $n^2$ tasks, each responsible for calculating one element of the result matrix $\mathbf{5}$. In this case, each task needs one row of the first input matrix and one
column of second input matrix to compute its result. However, it does not need to communicate with any other task. All \( n^2 \) tasks can function independently. [20] reports 776 to 11183-fold speedups using CUDA on a GPU for 4096X4096 matrices in comparison with a traditional sequential code on a single CPU core, using a range of improvements and parameter tunings including tile/thread-block dimensions, thread occupancy on cores, computational load per thread, and hiding memory latency between device memory shared by all streaming multiprocessors and smaller memory shared by 8 cores within a multiprocessor considering number/volume of memory transactions, coalesced access, bandwidth utilization, caching, bank conflicts, register usage, etc. These GPU implementations correspond to 27 to 407-fold speedups relative to a 4-thread OpenMP cache-optimized code on the multicore CPU. These experiments are reported for a 2.8GHz Xeon quad-core CPU and 240-core NVIDIA Tesla C1060 GPU with 8 streaming multiprocessors each with 30 cores. Another example is searching big files to find specific phrases. As the searching process over different part of one file or over several files is independent, the search operation can be done in parallel; concurrent tasks do not require communication with each other at all. Sifting through large volume shape files to find specific objects has been extensively used in GIS applications [14].

Some fundamental building blocks of data mining algorithms can be formulated as embarrassingly parallel problem. Some examples in this category are finding area in a region, generating minimal bounding rectangles of individual polygons, points in polygon test, update phases of k means clustering, etc. In points in polygon test [15] for a set of given points and a polygon, the membership of each point in the given polygon is to be determined. Membership status of each point is independent of the status of other points, making it embarrassingly parallel. Similarly, the problem of forming k clusters in a given GIS grid is embarrassingly parallel. Initially \( m \) random points are selected as cluster centers and each center is moved to mean value of distances of \( k \) nearest neighbors from each center. These \( m \) centers are updated iteratively. New mean value of each cluster depends only on old mean value of that cluster; hence the problem is task and data independent ([10] reports 13X speedup on NVIDIA GeForce 8800 GTX Ultra with 128 cores). Most pixel-based or fixed neighborhood based computations over raster data such as intersection over multiple data layers and simple stencil-based image processing tasks fall into this category ([4] reports 169 fold speedup for bilateral filtering on Nvidias GTX 280 GPU with 240 cores). It is also easy to implement these embarrassingly parallel problems using map-reduce paradigm, wherein the intermediate results calculated by each slave nodes are conveyed to a master node.([40] reports linear speedup in a distributed cluster for k means clustering.)

**Hotspot detection over a fixed neighborhood:** Hotspots are regions exhibiting unusual aggregation or outcome of a phenomenon. Some examples of hotspot detection are detecting aggregation of some epidemic, snowfall prediction, detecting most profitable regions to open a store for a business, regions of high level of radiations on a planet, etc. In simple formulations, hotspot detection depends on the value of the point in relation to its neighbors. In mathematical terms, on a given domain of data set, each point of the domain holds a value of discrete/continuous random variable. The value of the random variable is compared to the extent of a fixed radius using some predefined algorithm, and based on these comparisons, a spatial pattern of its distribution is determined. The hotspots are the regions exhibiting the desired values/shapes of this pattern [27]. This problem is embarrassingly parallel because usually each unit of the domain is processed with the same algorithm and is independent of the rest of the data set outside its radius of comparison (although, not all formulations are embarrassingly parallel). [21] reports up to 400X speedup on NVIDIA GTX280 GPUs.

### 3 Regular Computations

We now introduce the regularly structured problems and data which are the first tier of what is considered non-trivial parallelization. Some raster data computation, complex matrix manipulation problems, problems with many data independent loops, and certain types of graph problems typically fall within this category. Paral-
Parallelization is achieved by exploiting the independent nature of the instructions and/or the data in order to achieve good speedup. Matrix manipulation and raster data in particular lend themselves very well to GPU computation due to their reliance on basic data types and array-based structures. Graph algorithms can be performed on GPUs typically using their adjacency matrix representations which can then be stored conveniently on the GPU [18]. All-to-all Floyd-Warshall shortest path algorithm or Gaussian elimination for a system of linear equations, for example, manifest regular pattern of data access when parallelized [13].

The data collected in Earth Science are usually stored as raster data, where space and time are partitioned into regular grids (e.g., latitude and longitude, weeks/months/years) with attributes imposed on the grids (e.g., rainfall, vegetation cover index). Examples of such data include remotely sensed images of Earth surface at various resolutions [8, 24], ground observations of temperature, precipitation, etc. [16]. Pattern discovery on such datasets are potentially suitable for parallel computing due to the regular structures (grid) in the data. We introduce one example from our recent work, namely, the interesting sub-path/sub-region discovery problem [41].

**Interesting Sub-path/Region Discovery Problem**

Given a spatiotemporal (ST) dataset and a path embedded in its spatiotemporal framework, the goal of the interesting sub-path discovery problem is to identify all qualifying contiguous subsets of the given path according to the given interest measure and interestingness test. The ability to discover interesting sub-paths is important for many application domains. For example, vegetation cover is often used to study the response of ecological zones to climate change, which may vary across different ecological zones in the world. Given a path (e.g., along a longitude in Africa) and an interest measure of abrupt change, one can find sub-paths (e.g., the paths across the Sahel) with sharp increases (decreases) of vegetation cover. Such sub-paths may outline the spatial footprint of the transitional areas (known as ecotones [25]) between ecological zones. Due to their vulnerability to climate changes, finding and tracking ecotones gives us important information about how the ecosystem responds to climate changes. Figure 1(a) shows the Sahel region (highlighted area in the box), a well-known ecotone in Africa. The color represents the amount of vegetation measured in the normalized difference of vegetation index (NDVI). The vegetation cover along a spatial path (in red) is plotted in Figure 1(b) as a sequence. As can be observed, the sub-path between 15N and 20N latitude exhibit an abruptly decreasing trend. This reflects the transition from tropical savannah to desert.

![Vegetation cover in Africa](GIMMS Dataset [8, 36])

![Vegetation cover along highlighted longitudinal path](South to North)

![The footprint of Sahel discovered](Figure 1(c))

Figure 1: An example of interesting sub-path in vegetation cover data (best viewed in color).

In the specific application scenario described above, the goal is to discover all the sub-paths with abrupt increase/decrease along each longitude of the input dataset/image. Particularly, we require that an abrupt change sub-path should not be a subset of others so that redundant results could be removed. By performing this analysis on each longitudinal column of the input data image, we are able to outline the entire Sahel region by combining the footprint of Sahel along each longitudinal band. Figure 1(c) shows the output of the problem.
with \( \text{min}_\text{change} = \) the 90th percentile of all the longitudinal unit changes, where maroon and aqua represent decreases and increases of vegetation cover from south to north, respectively.

**Sequential and GPU Solutions**

The sequential solution to the above problem is non-trivial to design in that (1) the patterns lack of monotonicity (decreasing segments in a increasing sub-path), (2) Unknown maximum length of change ranging from 0 to the entire path, and (3) large volume of input data. A naive solution to this problem has two steps. Step 1: enumerate all the sub-paths in each longitudinal path in the dataset and test the interest measure function to find a list of candidate patterns. Step 2: enumerate every pair of sub-paths in the candidate list to eliminate dominated (i.e., subset) sub-paths. This solution has high time cost. For a single input longitudinal path with \( n \) locations, the time complexity reaches \( O(n^4) \) in the worst case, bringing the total time complexity on a data image with \( m \) such longitudinal columns to \( O(m \cdot n^4) \) in the worst case.

In our previous work, a sub-path enumeration and pruning (SEP) approach was proposed to improve the performance. First, a linear scan of all the units in the input data path is performed to precompute the basic function needed in the interest measure and store them in a set of lookup tables. This step allows a constant-time evaluation of each sub-path. Second, a row-wise traversal order is used to enumerate all the sub-paths from longer ones to short ones. Once a sub-path is identified as an interesting sub-path, all its subsets will be pruned from the enumeration to avoid further cost. Finally, the algorithm filters dominant sub-paths in the candidate list. The SEP row-wise algorithm takes \( O(n) \) (to build the lookup table) in best scenario and \( O(n^2) \) in the worse case. For an entire input image \( (m \times n) \), the time complexity is reduced to \( O(m \cdot n^2) \).

**GPU Implementation:** The GPU implementation of the interesting subpath problem is quite straightforward and leverages the fact that each worst-case possible path is data independent from every other query and each thread is largely performing the exact same set of computations. We can perform the naive algorithm in parallel by launching a thread for every possible interesting interval, in effect eliminating a theoretical order of magnitude from the total complexity. In many respects this gives a very nice speedup, however the SEP algorithm was found to be roughly 50 times faster [28].

Implementing the SEP algorithm on the GPU required some re-engineering in order to accommodate concepts that do not translate nicely to the GPU architecture such as dynamic data structures [26] and code branching. However, as the overall behavior of the algorithm is quite regular it produced very nice results that, with some constraints relaxed, allowed for near real-time computation and rendering of results to be achieved [28]. Instead of naively launching a thread for every conceivable outcome only half as many threads were launched. These threads were launched more intelligently with kernel dimensions that limited the potential effects of branching while also coalescing global memory read and write access. This also allowed leveraging shared memory on the GPU which drastically lowered compute time by reducing the time necessary for data access. Overall this sped up computation from 836ms for sequential Row-wise SEP to 35.2ms on the GPU for a single raster image (finding all intervals in the image), and from 576s to 20.1s for the entire raster image dataset.

This particular problem lent itself very well to implementation on the GPU as the dataset was raster data [8] and the individual pieces of the total dataset were rather small (2 MB). This allowed further parallelization to be undertaken. Each image being independent allowed us to further distribute this problem using the Hadoop framework across multiple GPU device nodes and achieve a near linear speedup for each additional node that was added. It also allowed us to run multiple instances of the program on a single GPU, however this is much harder as the intermediary data gets quite expensive to hold in memory and the GPU (at the time of experiment, NVidia GTX480) is resource limited to roughly 1 GB of memory on a standard consumer grade GPU.
4 Irregular Computations

GPUs are very effective at exploiting parallelism in regular, data-parallel computations, and we have seen some examples of efficient GPU parallelizations. However, many algorithms have to build, traverse, and update irregular data structures such as trees, graphs, and priority queues to solve a given problem. Examples of irregular computations includes breadth-first search, single-source shortest paths, n-body simulation, etc. In the context of spatial algorithms, spatial overlay and join are examples of irregular computation. In these examples, accesses to tree/graph based data structure is unpredictable and data-dependent. Moreover, parallelization of these data structure is hard because of the underlying tree topology and the fine-grained computation.

Spatial overlay is the process of interrelating several spatial features (points, lines, or polygons) from multiple datasets or layers of data, which creates a new output layer, visually similar to stacking several maps of the same region together. Spatial overlay has been accelerated by NVIDIA Tesla GPU in [7, 22]. McKenney et al. [22] developed GPU implementation of line segment intersection and the arrangement problem for overlay computation. For some datasets, authors show that their implementation of geospatial overlay on GPU runs faster than CPU-based plane-sweep implementation. Audet et al. [7] developed CUDA implementation of Uniform Grid based overlay algorithm originally designed by Franklin et al. [12].

Spatial Join is a type of table join operation in which fields from one layer’s attribute table are appended to another layer’s attribute table based on the relative locations of the features in the two layers. A typical example of a spatial join is “Find all pair of rivers and cities that intersect.” Recent work on GPU based parallel spatial join are discussed in [39, 33, 38, 6, 34]. Authors in [38] describe how a spatial join operation with R-Tree can be implemented on a GPU. Hadoop map-reduce based spatial join algorithm originally designed by Franklin et al. [12]. Polygonal overlay

Arguably, polygonal overlay computation is one of the difficult computations due to irregular distribution of arbitrary shaped polygons. Our Azure cloud based parallel system for polygon overlay known as Crayons is described in [1, 3]. It achieves 90x speedup for its task processing phase on Azure cloud - an embarrassingly parallel phase once pairs of potentially intersecting polygons have been identified constituting independent tasks. However, the end-to-end speed up is only 3 fold. An MPI version of Crayons is described in [2]. Polygon overlay processing algorithms using Hadoop map-reduce framework are described in [30], with three versions depending on the nature of the datasets, yielding up to 10x speedup. GPU-based parallelization of key tree-based data structures, namely R-tree [29], with 200 fold speedup for construction, and heap [17], with 40X speedup for concurrent inserts and min-deletes, have been explored recently. The parallel R-tree has been employed in MPI-GIS, our polygon overlay system, which yields 40x to 70x speedup in comparison to ArcGIS [28]. The architecture of MPI-GIS is shown in Figure 2a. Figure 2b shows the execution time for overlaying two real-world GIS datasets.

5 Conclusions

We presented three classes of computations, namely embarrassingly parallel, regular and irregular computations, and illustrated each with representative parallelization of GIS computations and analytics over GPUs.

References

Figure 2: (a) Architecture of MPI-GIS for Polygon Overlay using MPI + Pthread + GPU. (b) Execution time of MPI-GIS with varying number of MPI processes for overlaying USA Detailed Water Bodies and USA Block Group Boundaries on a cluster with 32 compute nodes having 8 cores/node


[38] T. Yampaka and P. Chongstitvatana. Spatial join with R-tree on graphics processing units, IC2IT, 2012.
Large-Scale Spatial Data Processing on GPUs and GPU-Accelerated Clusters*

Jianting Zhang1,2, Simin You2, Le Gruenwald3
1Department of Computer Science, City College of New York, USA
2Department of Computer Science, CUNY Graduate Center, USA
3School of Computer Science, University of Oklahoma, USA

Abstract

The massive data parallel computing power provided by inexpensive commodity Graphics Processing Units (GPUs) makes large-scale spatial data processing on GPUs and GPU-accelerated clusters attractive from both a research and practical perspective. In this article, we report our works on data parallel designs of spatial indexing, spatial joins and several other spatial operations, including polygon rasterization, polygon decomposition and point interpolation. The data parallel designs are further scaled out to distributed computing nodes by integrating single-node GPU implementations with High-Performance Computing (HPC) toolset and the new generation in-memory Big Data systems such as Cloudera Impala. In addition to introducing GPGPU computing background and outlining data parallel designs for spatial operations, references to individual works are provided as a summary chart for interested readers to follow more details on designs, implementations and performance evaluations.

Keywords: Spatial Data, Large-Scale, Data Parallel Design, GPGPU

1 Introduction

Geospatial data is one of the fast growing types of data due to the advances of sensing and navigation technologies and newly emerging applications. First of all, the ever-increasing spatial, temporal and spectral resolutions of satellite imagery data have led to exponential growth of data volumes. Second, both airborne and mobile radar/lidar sensors have generated huge amounts of point-cloud data with rich structural information embedded. Third, many mobile devices are now equipped with locating and navigation capabilities by using GPS, cellular and Wifi network technologies or their combinations. Considering the large amounts of mobile devices and their users, the accumulated GPS traces, which are essential to understand human mobility, urban dynamics and social interactions, can be equally computing demanding when compared with satellite imagery data and lidar point cloud data. While the traditional infrastructure data, such as administrative regions, census blocks and transportation networks, remain relatively stable in growth when compared with the new types of geospatial data, quite often the new sensing and location data need to be related to the infrastructure data in order to make sense out of them. Furthermore, polygons derived from point data clustering (e.g., lidar point clouds, GPS locations) and raster data segmentations (e.g., satellite and airborne remote sensing imagery) are likely to be even larger in volumes and computing-intensive. To efficiently process these large-scale, dynamic and diverse geospatial data and to effectively transform them into knowledge, a whole new set of data processing techniques are thus required.

Existing Big Data techniques include algorithmic improvements to reduce computation complexity, developing approximate algorithms to trade accuracy with efficiency and using parallel and distributed hardware and systems. As parallel hardware, such as multi-core CPUs and many-core Graphics Processing Units (GPUs), is now mainstream commodity [4], parallel and distributed computing techniques on top of the inexpensive commodity hardware are attractive, especially for applications that require exact computation while little room has been left for algorithmic improvements. In the past few years, the simplicity of the MapReduce computing model and its support in the open source Hadoop system have made

*Partially supported by NSF Grants IIS-1302423 and IIS-1302439.
it attractive to develop distributed geospatial computing techniques on top of MapReduce/Hadoop [2]. The success of SpatialHadoop [3] and HadoopGIS [1] has demonstrated the effectiveness of MapReduce-based techniques for large-scale geospatial data management where parallelisms are typically identified at the spatial partition level which allows adapting traditional serial algorithms and implementations within a partition.

While MapReduce/Hadoop based techniques are mostly designed for distributed computing nodes each with one or multiple CPU cores, the General Purpose computing on Graphics Processing Units (GPGPUs) techniques represent a significantly different parallel computing scheme. GPU hardware architectures adopt a shared-memory architecture closely resembles traditional supercomputers [5], which requires fine-grained thread level coordination for data parallelization. From a practical perspective, as the data communications are becoming increasingly expensive when compared with computation on modern processors/systems [4], GPU shared-memory architectures allow very fast data communications (currently up to 672 GB/s for Nvidia GTX Titan Z\footnote{http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-titan-z}) among processing units when compared with cluster computing (50 MB/s in cloud computing and a few GB/s in grid computing with dedicated high-speed interconnection networks) and multi-core CPUs (a few tens of GB/s), which is desirable for data intensive computing. Finally, in addition to fast floating point computing power and energy efficiency, the large number of processing cores on a single GPU device (5,760 for Nvidia GTX Titan Z) makes it ideal for processing geospatial data which is typically both data-intensive and compute-intensive. Nevertheless, from a research perspective, techniques based on a single GPU device have limited scalability which makes it desirable to scale-out the techniques to cluster computers with multiple-nodes and multiple GPU devices.

In this paper, we report our work on data parallel designs for several geospatial data processing techniques. By further integrating these GPU-based techniques with distributed computing tools, including Message Passing Interface (MPI\footnote{http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-titan-z}) library in the traditional High-Performance Computing (HPC) clusters and newer generation of Big Data systems (such as Impala\footnote{http://impala.io} and Spark\footnote{http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-titan-z}) for Cloud computing, we are able to scale the data parallel geospatial processing techniques to cluster computers with good scalability. While we are aware of the complexities in developing a full-fledged GIS and/or a Spatial Database on GPUs, our research bears three goals: 1) to demonstrate the feasibility and efficiency of GPU-based geospatial processing, especially for large-scale data, 2) to develop modules for major geospatial data types and operations that can be directly applied to popular practical applications, such as large-scale taxi trip data and trajectory data, and 3) to develop a framework to integrate multiple GPU-based geospatial processing modules into an open system that can be shared by the community. We have developed several modules (as summarized in Fig. 4 in Section 3), over the past few years. We are in the process of integrating these modules under a unified framework and developing new modules to further enhance functionality. Interested readers can follow the respective references for more details.

For the rest of the paper, Section 2 provides a brief introduction to GPGPU computing; Section 3 introduces our data parallel designs and GPU implementations; Section 4 presents the high-level designs and implementations on integrating single-node GPU techniques for scaling out geospatial processing on GPU-accelerated clusters; and finally Section 5 is the summary and future work directions.

## 2 GPGPU Computing


Modern GPUs are now capable of general computing [4]. Due to the popularity of the Compute Unified Device Architecture (CUDA) [6] on Nvidia GPUs, which can be considered as a C/C++ extension, we will mostly follow CUDA terminologies to introduce GPU computing. Current generations of GPUs are used as accelerators of CPUs and data are transferred between CPUs and GPUs through PCI-E buses. The Nvidia Tesla K10 GPU shown in the top-right side of Fig. 1 has 15 Streaming Multiprocessors (SMXs) with each SMX having 192 processing cores. Since 32 processing cores form a warp and warps are used as the basic units for scheduling, GPUs can be viewed as Single Instruction, Multiple Data (SIMD) devices [4]. A multiprocessor can accommodate multiple thread blocks with each thread block having one or more warps through time multiplexing to hide I/Os and other types of latencies. For example, Tesla K10 GPU supports up to 2048 concurrent threads (i.e., 64 warps) per SMX. All the 32 threads in a warp execute the same instruction and the performance is maximized when there are no code branches within the warp; otherwise the branches will be serialized and the performance can be poor.

GPUs that are capable of general computing are facilitated with Software Development Toolkits (SDKs) provided by hardware vendors. The left side of Fig. 1 shows a simple example on summing up two vectors (A and B) into a new vector

---

two vectors ($A$, $B$) provided by hardware vendors. The left side of Fig. 1 shows a simple example on summing up

The whole computing task is divided into $M$ blocks with each being assigned to a thread block with $N$ threads. Within a thread block, an index can be computed to address the relevant vector elements for inputs/outputs based on its thread identifier ($threadIdx.x$) and block identifier ($blockIdx.x$), which are automatically assigned by the hardware scheduler, and block dimension ($blockDim.x$) which is specified when the kernel is invoked. While we use a 1D example in Fig. 1, CUDA supports up to three dimensions.

Parallelism is fundamental to data processing on parallel hardware. While coarse-grained parallelization can be used to create parallel tasks and exploit existing scheduling algorithms for parallel execution, the reverse is not true. Roughly speaking, the CUDA computing model for GPUs supports both task parallelism at the thread block level and data parallelism at the thread level. For a single GPU kernel designed for solving a particular problem, the boundary between task and data parallelism can be configured when the kernel is invoked (the lower-left part of Fig. 1). However, to maximize performance, data items should be grouped into basic units that can be processed by a warp of threads (which are dynamically assigned to processor cores) without incurring significant divergence. Instead of accessing data items sequentially that exhibits significant temporal locality that is optimal on CPUs, when nearby threads in a warp access a continuous block of data items in GPU device memory, the individual GPU memory accesses by the warp of threads can be combined into fewer memory accesses (coalesced memory accesses). This GPU characteristic requires a careful design of the layouts of multi-dimensional spatial data structures and their access patterns when developing spatial algorithms.

The unique hardware features and large tunable parameter space have made developing efficient GPU programs challenging. Using local, focal, zonal and global classification of geospatial operations [11] for both vector and raster data, as local operations only involve independent individual data items and focal operations mostly involve a bounded small number of neighboring items, they are relatively easy to be parallelized on GPUs. However, zonal operations (such as generating elevation distribution histograms for raster cells in polygons) and global operations (such as indexing vector geometry as trees) typically involve geometrical objects with variable numbers of vertices and may be spatially related to unbounded numbers of geometrical objects, such as joining two polygon datasets based on point-polygon test or two polyline datasets based on distance or similarity measures. The irregularities of data layout and data access patterns in such spatial operations have made it technically very challenging to design and implement efficient geospatial algorithms on GPU hardware.

While we have developed some geometrical algorithms on GPUs using CUDA directly at the beginning of our explorations of massive data parallel computing power for geospatial processing, we gradually realized that the straightforward approach is not productive. Instead, we have chosen to adopt a parallel primitive based approach whereas possible to reduce implementation complexity and improve development productivity. Parallel primitives refer to a collection of fundamental algorithms that can run on parallel machines [8]. The behaviors of popular parallel primitives on 1D arrays are well-understood. Parallel primitives usually are implemented on top of native parallel programming languages (such as CUDA) but provide a set of simple yet powerful interfaces (or APIs) to end users. Technical details are hidden from end users and many parameters that are required by native programming languages are fine-tuned for typical applications in parallel libraries so that users do not need to specify such parameters explicitly.
3 Data Parallel Designs and Single-Node GPU-Implementations

Due to space limit, we will use a grid-file based indexing as an example to illustrate the idea of parallel primitives based data parallel designs and their implementations on GPUs. We then provide a summary chart for our existing designs and implementations and refer the readers to respective references for details.

Consider indexing a large set of points using the classic grid-file structure [9]. While serial algorithms and their implementations loop through all the points and determine the grid cell that each point should be associated with, as shown in Fig. 2, we use four parallel primitives for this purpose: transform, (stable) sort, reduce (by key) and (exclusive) scan. The transform primitive (similar to the Map function in MapReduce) generates Morton codes [9] that are used as grid cell identifiers for all points at a pre-defined resolution level; the sort primitive sorts points based on the cell IDs; the reduce (by key) primitive counts the number of points within each grid cell; and finally the (exclusive) scan primitive computes the prefix-sums of the numbers of points in all grid cells which are the starting positions of the points in the sorted point data vector. The primitives are executed by GPU hardware in parallel using their most efficient implementations which are transparent to algorithm and application developers. In fact, the current Thrust\(^5\) parallel library (which comes with CUDA SDK) uses radix sort for the sort primitive. Although quicksort is known to be efficient on CPUs, radix sort is considered to be more efficient on GPUs.

![Diagram of parallel primitives](image)

Figure 2: Data Parallel Design and Implementation of Grid-File Point Indexing on GPUs

We have designed indexing techniques for rasters [21, 22, 19], points [18, 26] and Minimum Bounding Boxes [26, 13] using Grid-Files [26], Quadtrees [18, 21, 22, 19] and R-Trees [13]. We have also developed a GPU-based spatial join framework to join two indexed spatial datasets based on point-in-polygon tests [18], point-to-polyline distance [26], polyline-to-polyline similarity [23] with applications to spatiotemporal aggregation of large-scale taxi-trip data [18], trip-purpose analysis [25], trajectory similarity query [23] and global biodiversity studies [20]. Fig. 3 illustrates our framework for spatial join processing on GPUs using grid-file indexing. After MBRs are rasterized into grid cells, the middle part of Fig. 3 illustrates how to use parallel primitives including sort, binary searches and unique to pair up polygons or polylines (i.e., spatial filtering) for the subsequent spatial refinement. While using quadtrees or R-trees for spatial filtering may require different parallel primitives (we refer to [18, 13] for details), the grid-file based spatial filtering essentially transforms a spatial query (filtering) problem into a relational equi-join problem which has been shown to be effective on GPUs [26]. The lower part of Fig. 3 shows four types of spatial refinement operations which can be realized efficiently on GPUs and we refer to [18, 26, 25] for details.

In addition to spatial indexing and query processing, which are important components in spatial databases, our research also involves several modules that are more related to pre-processing and post-processing as well as data conversions on GPUs, which are essential in a GIS environment. The work on natural neighbor based spatial interpolation for lidar data [12], although is implemented using CUDA directly for performance, also adopts a parallel primitive approach internally at the thread block level. The spatial interpolation module naturally bridges point data and raster data which makes it possible to apply existing techniques for rasters for point data. Similarly the GPU-based polygon rasterization technique in [16] bridges polygons and rasters. In observing that indexing polylines and polygons at MBR level might be limited by high false positives and result in low indexing power, we have developed a polygon decomposition technique which can decompose polygons into quadrants [24]. The decomposed polygons can be used for both indexing and approximating polygons in certain queries. We have also performed preliminary designs and implementations for polygon overlays [28], Geographical Weighted Regression (GWR) analysis [15] and map-matching for trajectories on GPUs. Fig. 4 provides

---

\(^5\)https://thrust.github.io
a summary chart of our existing works where shaded rectangles represent indexing techniques, diamond-headed edges represent spatial applications and bracketed numbers represent publication sources for more details.

Figure 3: A Framework of Spatial Join Query Processing on GPUs using Grid-File Indexing

4 Scaling-out to GPU-Accelerated Clusters

To further improve the performance of large-scale geospatial data processing, it is essential to share workloads among distributed computing nodes that are equipped with GPUs for scalability. As discussed in Section 1, it is nontrivial to design and implement efficient distributed computing systems while existing Big Data systems typically do not support spatial data processing. In observing that improving single-node efficiency using GPUs can significantly reduce inter-node data communications [10], we believe that integrating our single-node GPU-based geospatial processing techniques with distributed computing techniques can be competitive with existing solutions (such as HadoopGIS [3] and SpatialHadoop [1]).

Towards this goal, we have experimented two approaches: one using the MPI parallelization software stack available on the ORNL Titan supercomputer and one using the open source Cloudera Impala [7]. Fig. 5 illustrates the framework of the first approach where the NASA SRTM 30-meter DEM rasters with 20 billion raster cells are first divided into raster tiles and the tiles are paired up with county MBRs. The pairs are partitioned and the partitions are then sent to Titan computing nodes through MPI APIs (Left Fig. 5). On each computing node, a raster tile is further divided into blocks and
The second approach we have adopted is to extend Cloudera Impala to support spatial query in SQL. Different from traditional distributed computing that utilizes MPI, data communication in Impala is based on Apache Thrift and is tightly embedded into SQL physical execution plan. As shown in Fig. 6, in the ISP prototype system we have developed, three additional extensions are implemented in order to reuse the Impala infrastructure for distributed spatial query processing.

6http://thrift.apache.org
First, we modify the Abstract Syntax Tree (AST) module of Impala frontend to support spatial query syntax. Second, we represent geometry of spatial datasets as strings to support spatial data accesses in Impala and prepare necessary data structures for GPU-based spatial query processing. Third, we have developed a *SpatialJoin* module as a subclass of *ExecNode* to extract data from both left and right sides in a spatial join in batches before the data is sent to GPU for query processing. We again refer to our technical report [27] for details while only provide a summary of performance evaluation here due to space limit.

For future work, first of all, we would like to investigate on both generic and spatial specific parallel primitives for multi-dimension data as most of existing primitives in parallel libraries (including Thrust) are designed for one dimensional vectors. Second, while we have successfully scaled out our data parallel designs from a single node to distributed nodes, there is still considerable room for optimizations to further improve scalability by incorporating spatial processing domain semantics. Finally, compared with existing spatial databases that provide declarative SQL interfaces and GIS that provide intuitive graphics interfaces, except for ISP, most of our prototypes are standalone command line programs. We plan to integrate both SQL and graphics interfaces with our prototypes for better usability.

The current GPU SDKs have limited support for JAVA, Scala and other languages other than C/C++, which makes it difficult to integrate our GPU-based implementations with Hadoop and Spark for scalability. However, we have observed that our data parallel designs and their implementations on top of the Thrust parallel library have strong connections with the built-in vector functions (e.g., *map, reduce* and *sort*) in Scala (and similarly Java 8). The connections have motivated us to develop SpatialSpark [14] to process spatial queries directly on Spark, a popular and high-performance in-memory Big Data system developed using Scala and Java. While the end-to-end performance of SpatialSpark is largely affected by the underlying geometry library (JTS\(^7\) in this case) which dominates the spatial join query runtimes, the simple implementations and high-performance have made the implementation attractive for Cloud deployment [14]. This subsequently has motivated us to develop a data communication infrastructure similar to Spark (and Akka\(^8\) that Spark depends on) to natively support large-scale geospatial processing on GPU-accelerated clusters. By developing more semantics-aware spatial data partition and communication primitives and extending the row-batch based asynchronous data processing framework in Impala [7] to semi-structure data (such as spatial data and trajectory data), we hope the new designs and implementations can bring higher efficiency and scalability for large-scale geospatial processing.

5 Conclusion and Future Work

Large-scale geospatial data in newly emerging applications require new techniques and systems for better scientific inquiries and decision making. Efficient and scalable processing of large-scale geospatial data on parallel and distributed platforms is an important aspect of Big Data research. In this paper, we present our work on parallel designs and implementations of geospatial processing algorithms and systems on GPUs and GPU-accelerated clusters for both efficiency and scalability. Experiments on several large-scale geospatial data have demonstrated orders of magnitude speedups when compared with traditional techniques on single CPU cores and have shown great potentials in significantly speeding up a wide range of geospatial applications.

For future work, first of all, we would like to investigate on both generic and spatial specific parallel primitives for multi-dimension data as most of existing primitives in parallel libraries (including Thrust) are designed for one dimensional vectors. Second, while we have successfully scaled out our data parallel designs from a single node to distributed nodes, there is still considerable room for optimizations to further improve scalability by incorporating spatial processing domain semantics. Finally, compared with existing spatial databases that provide declarative SQL interfaces and GIS that provide intuitive graphics interfaces, except for ISP, most of our prototypes are standalone command line programs. We plan to integrate both SQL and graphics interfaces with our prototypes for better usability.

\(^7\)http://www.vividsolutions.com/jts/JTSHome.htm
\(^8\)http://akka.io
References

Emerging Trends in Monitoring Landscapes and Energy Infrastructures with Big Spatial Data

Budhendra Bhaduri¹, Dilip Patlolla¹, Ranga R. Vatsavai¹,², Anil Cheriyadat¹, Wei Lu¹, Rajasekar Karthik¹
¹Geographic Information Science & Technology, Oak Ridge National Laboratory, USA
²Department of Computer Science, North Carolina State University, USA

Abstract

Explosion of spatial data from satellite to citizen sensors has posed the critical challenge of Big Spatial Data integration, analysis, and visualization. This article focuses on research and development activities at Oak Ridge National Laboratory (ORNL) that are addressing end-user applications utilizing high performance computing based geospatial science and technology solutions to optimize the analysis, modeling, and multi-megapixel scale visualization of the geospatial data. Specifically we highlight recent developments and successes in the areas of high resolution settlement mapping, transportation and mobility analysis, and effective monitoring of biomass for energy and food security.

1 Introduction

Understanding change, in space and time, through visualization of landscape processes often provides the most effective tool for decision support. Current geoanalytics are limited in dealing with temporal dynamics that describe observed and/or predicted behaviors of entities i.e. physical and socioeconomic processes. End point based temporal analysis can capture events signified by broad changes in observable earth features; but are inadequate for revealing temporal characteristics of the processes as defined by the onset, frequency, and rate of changes in the features. Analysis of disparate and dynamic geographic data provides an effective component of an information extraction framework for multi-level reasoning, query, and extraction of geospatial-temporal features. With increasing temporal resolution of geographic data, there is a compelling motivation to couple the powerful modeling and analytical capability of a Geographic Information System (GIS) to perform spatial-temporal analysis and visualization on dynamic data streams. However, the challenge in processing large volumes of high resolution earth observation and simulation data by traditional GIS has been compounded by the drive towards real-time applications and decision support. Based on our experience of providing scientific and programmatic support for United States (U.S.) federal agencies, we discuss progress and challenges of some of the emerging computational approaches, including algorithms and high performance computing, illustrated with settlement mapping, population dynamics, and sustainable energy and mobility science. These examples provide an insight into the potential power and possibilities of spatiotemporal analysis and visualization for operational and policy decision support across numerous disciplinary issues.

2 Background

2.1 GIScience and technology at Oak Ridge National Laboratory

Originally established in 1943 as part of the Manhattan Project, today Oak Ridge National Laboratory is the largest U.S. Department of Energy (DOE) science and energy laboratory, conducting basic and applied research to deliver transformative solutions to compelling problems in energy and security. The Geographic Information Science and Technology (GIST) Group at the Oak Ridge National Laboratory has been a pioneer in the development, implementation, and application of systems, science, and technology for geographic information since 1969 - well before the advent of commercial GIS. In the
DOE national laboratory system, it is the only computational research group focused on advancing the theory and applications of geospatial science and technology. Today, the GIST group is an internationally renowned research & development organization of over sixty diverse scientific professionals who bring together a wide range of expertise ranging from GIS and remote sensing technologies to key domain sciences. We focus on developing innovative, knowledge discovery, and solutions to solve multidisciplinary and complex problems for energy, environment, and national and homeland security missions. In addition, we actively partner with leading academic institutions to collaborate with faculty members and engage students to support the national priority of research, education, and workforce development. Current research and development capabilities are focused around:

Population Distribution and Dynamics: Research in this area includes algorithm and model development relating to high-resolution spatiotemporal mapping of population, socio-economic indices, and population-infrastructure dependencies.

Geographic Data Sciences: Research in this area encompasses geographic data analysis, spatiotemporal data mining to explore spatiotemporal database capabilities with management of uncertainties.

High Performance Geocomputation: Research in this area combines the strengths of geographical methods with computer and computational sciences to address data exploration through mining and visualization techniques that are beyond the present capabilities of common desktop computing using commercial GIS.

Research and development activities span across high resolution population distribution and dynamics, energy assurance including modeling and visualization of the electric grid and biomass and bioenergy resources, climate change science including climate extremes and integrated impact analysis, emergency and disaster management, critical infrastructure modeling and simulation, and earth science informatics. Our work has been supported by the U.S. Departments of Energy; Department of Homeland Security; Department of Defense; Department of Transportation; Environmental Protection Agency; Bureau of Census; National Aeronautics and Space Administration; National Institute of Health; National Oceanic and Atmospheric Administration; and a number of member agencies of the Intelligence Community.

2.2 Big Data and geospatial computing

Confirming the notion that “a picture is worth a thousand words”, the value of geospatial analysis and visualization of our environment has been well recognized and consequently geospatial data and models have become a critical part of the decision making process in planning, policy, and operational missions for government agencies from local to global scales. At present, we are climbing up the ledge of the data canyon, which is increasingly being flooded, at the rate of terabytes to petabytes of data a day, with geospatial data from earth observation and simulation. Current Geographic Information Systems, designed for geospatial integration and visualization, are of limited value in utilizing high performance computing and data assets that are required to advance our scientific understanding of earth system models and to provide decision support systems for time-critical missions. Oak Ridge National Laboratory is utilizing high performance computing based geospatial science and technology solutions to optimize the analysis, modeling, and multi-megapixel scale visualization of the geospatial data to foster development and integration of end-user applications. Increasing spatial and temporal resolutions of remotely sensed data have significantly enhanced the quality of mapping and change data products. Richness of pixel content has also provided the opportunity to develop novel data sets using machine learning and pattern recognition approaches. However, even with automation of such analysis on evolving computing platforms, rates of data processing have been suboptimal largely because of the ever-increasing pixel to processor ratio coupled with limitations of the computing architectures. Such constraint has posed a significant challenge for advancing the practice of change detection to continuous change monitoring, a much needed capability for developing disaster early warning or alerting systems.
3 Highlights of Recent Progress

In this section, we exemplify addressing the Big Spatial Data challenge with three broad research initiatives at Oak Ridge National Laboratory: (a) mapping global settlement and population distributions; (b) developing urban mobility models and simulations for energy; and (c) large scale monitoring of biomass for food and energy assurance.

3.1 Settlement mapping for population distribution

Urban and rural population distribution data are fundamental to prevent and reduce disaster risk, eliminate poverty and foster sustainable development. Commonly available population data, collected through modern censuses, do not capture this high-resolution population distribution and dynamics. Since late 1990s, ORNL has been developing LandScan Global (~1km resolution) population distribution data [1] and, since early 2000s, LandScan USA (~90m resolution) [2, 3] population dynamics (nighttime and daytime) databases. At such resolutions, LandScan Global and USA population databases are the highest resolution global and U.S. population distribution data available today and considered the community standard for estimating population at risk. We are currently developing LandScan HD, which, at ~90m cells, will improve the positional accuracy and relative density of population distributions globally by two orders of magnitude.

Geospatial data and models offer novel approaches to disaggregate Census data to finer spatial units; with land use and land cover (LULC) data being the primary driver. With increasing availability of LULC data from satellite remote sensing, “developed” pixels have been nucleus to assessing settlement build up from human activity. With the availability of moderate to high resolution LULC data derived from the National Aeronautics and Space Administration, Moderate Resolution Imaging Spectroradiometer (NASA MODIS) (250-500m) or Landsat Thematic Mapper (TM) (30m) have facilitated the development of population distribution data at a higher spatial resolution including LandScan Global and LandScan USA. Although these LULC data sets have somewhat alleviated the difficulty for population distribution models, in order to assess the true magnitude and extent of the human footprint, it is critical to understand the distribution and relationships of the small and medium-sized human settlements. These structures remain mostly undetectable from medium resolution satellite derived LULC data. For humanitarian missions, the truly vulnerable, such as those living in refugee camps, informal settlements and slums need to be effectively and comprehensively captured in our global understanding. This is particularly true in suburban and rural areas, where the population is dispersed to a greater degree than in urban areas.

Extracting settlement information from very high-resolution (1m or finer), peta-scale earth observation imagery has been a promising pathway for rapid estimation and revision of settlement and population distribution data. As early as 2005, automated feature extraction algorithms implemented on available Central Processing Unit (CPU)-based architectures demonstrated radical improvement in image analysis efficiency when manual settlement identification from a 100-km² area was reduced from 10 hours to 30 minutes [4]. However, this scaled inefficiently beyond 10 nodes and at that rate processing 57-million km² habitable area would take decades [5]. Most existing approaches used to classify very high-resolution (VHR) images are single instance (or pixel-based) learning algorithms, which are inadequate for analyzing VHR imagery, as single pixels do not contain sufficient contextual information. However, spatial contextual information can be captured via feature extraction and/or through newer machine learning algorithms in order to extract complex spatial patterns that distinguish between informal and formal settlements. In recent years, we made significant progress in advancing the state of art in both directions.

Our approach has been to focus on image patches instead of single pixels as unit of analysis. Patches (or objects) can be generated from segmentation or through fixed size tiling (or grids). Features extracted from these patches can be then be fed to the standard single instance learning algorithms for categorizing various settlements. However, given the challenges associated with segmentation and limitations of single
instance learning schemes, we also developed fixed size tile and multiple instance learning algorithms which shown to perform better in distinguishing informal settlements from formal settlements. The four components of our settlement mapping framework is shown in Figure 1.

**Figure 1:** Architecture of very high resolution imagery based settlement mapping system

*Segmentation* is widely used to generate image patches. Recent work [6] shows implementation and evaluation of the state of art segmentation methods. Performance benchmarking of seven segmentation algorithms against a comprehensive aerial image data set with human generated segmentations shows that graph-based region merging algorithm (Felz-Hutt), oriented watershed transform ultrametric contour maps with globalPb as contour detector (gPb-owt-ucm), and factorization-based segmentation algorithm give better performance than the others based on quantitative measures and visual inspection. Performance of segmentation algorithms varies significantly based on user defined input parameters, resulting in either over or under segmentations. Once segments are generated, one can extract object features like area, perimeter, and compactness.

*Features* that capture spatial and structural properties are very useful in identifying complex spatial patterns. We developed a large number of feature extraction techniques and evaluated the usefulness of these features in VHR image based settlement mapping. These features are summarized in Table 1, more details can be found in [7, 8]. Number of features generated depends on number of input bands, scales and orientations.

<table>
<thead>
<tr>
<th>Type of Feature(s)</th>
<th># Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLCM PanTex</td>
<td>3</td>
</tr>
<tr>
<td>Histogram of Gradients</td>
<td>15</td>
</tr>
<tr>
<td>Lacunarity</td>
<td>3</td>
</tr>
<tr>
<td>Linear Feature Distribution</td>
<td>6</td>
</tr>
<tr>
<td>Line Support Regions</td>
<td>9</td>
</tr>
<tr>
<td>Vegetation Indices</td>
<td>2</td>
</tr>
<tr>
<td>SIFT</td>
<td>96</td>
</tr>
<tr>
<td>TEXTONS</td>
<td>96</td>
</tr>
</tbody>
</table>

Table 1: Feature extraction methods
Classification testing and training includes several pixel-based (single instance) learning methods and as well as patch (object) based (multiple instance) learning techniques for settlement mapping. For patch based methods we used fixed size tile (grids) scheme to generate features for multiple instance learning (MIL) algorithms, which was found to perform better than single instance learning (SIL). Single instance learning algorithms are appropriate for thematic classification (e.g., roads, buildings), whereas multi-instance learning algorithms are designed for recognizing complex patterns (e.g., informal and formal settlements). Key idea behind multi-instance learning schemes is the utilization of all instances drawn from the image patches or windows. Our Citation-KNN based MIL approach where, a new patch is compared against all training patches by minimizing pair-wise distances between individual instances across the patches, was applied to DigitalGlobe CitySphere imagery (0.6m, 3 bands) for Caracas, Kabul, Kandahar, La Paz, and Accra. In case of patch-based classification we chose: formal and informal settlements, water, forest, and bare soils. Overall accuracy and advantages of MIL classification are discussed in details elsewhere [8] and results are summarized in the Table 2.

<table>
<thead>
<tr>
<th>City</th>
<th>Citation-kNN</th>
<th>Regression</th>
<th>RF</th>
<th>NN(MLP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accra</td>
<td>75.25</td>
<td>71.25</td>
<td>72.08</td>
<td>69.58</td>
</tr>
<tr>
<td>Caracas</td>
<td>82.96</td>
<td>78.15</td>
<td>81.85</td>
<td>81.81</td>
</tr>
<tr>
<td>La Paz</td>
<td>80.97</td>
<td>77.17</td>
<td>78.26</td>
<td>80.23</td>
</tr>
<tr>
<td>Kandahar</td>
<td>79.78</td>
<td>64.89</td>
<td>69.14</td>
<td>73.93</td>
</tr>
</tbody>
</table>

Table 2: Overall accuracy of MIL against SIL methods

Making informed national-level decisions based on image-derived analytics require processing petabytes of high-resolution satellite image data. The computational process often involves extracting, representing, and classifying visual data patterns to detect different types of man-made structures [8]. Computing numerical feature descriptors from raw image data is an expensive but key step in the computational process. Human settlement maps extracted from high resolution (0.6m) satellite imagery serves as a foundational input to the population model. The pixel to feature descriptor mapping is an expensive, but a critical step, in automated scene analysis. At that resolution, analyzing the global land surface (~150 Million km$^2$ or ~600 trillion pixels) utilizing settlement detection algorithms on petascale image databases easily saturates the processing capabilities of conventional CPUs. Our approach of parallel feature extraction, based on implementation of two different feature representation techniques – (i) Textons and (ii) Dense Scale Invariant Feature Transforms (DSIFT), for large-scale image analysis has achieved unprecedented speedup (10X-200X speedup depending on feature) as compared to CPU and overall settlement detection accuracy (over 85%) [9]. With this computational framework processed sub-meter spatial resolution aerial imagery at the rate of 22 sq. km / sec [10]. Figure X shows an example from Addis Ababa, Ethiopia where, for an area of 800 km$^2$, settlements were mapped with 93% accuracy in less than 18 seconds on a graphics processing unit (GPU) based platform.

Figure 2: Settlements (in red) extracted from very high resolution satellite images for Addis Ababa, Ethiopia
High performance computing (HPC) based settlement mapping has been extended to automated identification of mobile home parks in the 48 U.S. states using pattern recognition and machine learning techniques. Mobile home parks present a challenging class of scenes for automated detection from satellite images given their large within class variations. ORNL’s approach for automated scanning and characterizing the settlement regions to detect mobile home park locations are producing >90% accuracy [11].

3.2 Urban mobility modeling for energy

Mobility is an important measure to evaluate the efficiency, accessibility, and livability in urban areas. Transportation simulations are frequently used for mobility assessment in urban planning, traffic operation, and emergency management. In order to reduce society’s dependence on fossil fuels, environmental impacts, and congestion, a number of alternative energy supply, distribution, and end-use transportation systems, technologies and policies are presently being explored. These include conventional hybrid vehicles, Plug-in Hybrid Electric Vehicles (PHEVs), increased usage of biofuels, and connected and automated vehicles. Ideally, development and implementation of future strategies for alternative energy resources and technologies will assure a societal system in which energy, environment, and mobility interests are simultaneously optimized. Given the complex, intertwined nature of such system across geographic scales, assessing the effectiveness of possible planning strategies and discovering their unanticipated and unintended consequences require modeling and simulation utilizing finest resolution data, physical and social processes, and observing the emerging behavior of the system over large spatial and temporal scales.

For knowledge discovery, characterization of the interaction between the human dynamics and transportation infrastructure is essential and requires integration of three distinct components, namely, data, models and computation. Recently, few models have started addressing the human dynamics of physical and social systems. However, none has been able to successfully integrate both the physical as well as behavioral aspects. Progress has also been limited by data and computational challenges necessary for accommodating the required high resolution along spatial, temporal and behavioral dimensions. Integration of high-resolution socio-demographic data and models bring much promise for capturing the social/behavioral dimension. This dimension is essential in enabling us to characterize the interplay and interdependencies between (transportation) technologies and societal features that are likely to: (i) have an impact on the success of future technologies and (ii) be overlooked by current approaches of modeling at aggregated scales. To judiciously evaluate the impacts of multiple transformational mobility/energy/environment optimization strategies there is a clear need to create a modeling and simulation framework of regional transportation processes with high-resolution geographic, demographic, socio-economic data and behavioral characteristics. No such modeling and simulation frameworks exist today. At Oak Ridge National Laboratory, we are combining the strengths of geospatial data sciences, high performance simulations, transportation planning, and vehicle and energy technology development to design and develop a national knowledge discovery framework to assist decision makers at all levels – local, state, regional, and federal [12, 13].

With increasing availability and quality of traditional and crowdsourced data, ORNL has utilized the OpenStreetMap roads network, and has integrated high-resolution population data with traffic simulation to create a Toolbox for Urban Mobility Simulations (TUMS) at global scale [14, 15]. Consistent data and simulation platform allows quick adaption to various geographic areas that has been demonstrated for multiple cities across the world. TUMS consists of three major components: data processing, traffic simulation models, and Internet-based visualizations (Figure X) [16].

It integrates OpenStreetMap, LandScan™ population, and other open data (Census Transportation Planning Products, National household Travel Survey, etc.) to generate both normal traffic operation and emergency evacuation scenarios [17]. TUMS integrates transportation analysis simulation system (TRANSIMS) and microscopic traffic simulator (MITSIM) as traffic simulation engines, which are open-
source and widely-accepted for scalable traffic simulations. The portal-based User Interface and two levels of visualization meet distinct community requirements. Link-based macroscopic visualization provides network performance analysis for planning. Vehicle-based microscopic visualization allows driving behavior monitoring and evaluation of engineered connected and automated vehicle system.

![Figure 3: Three primary components of the TUMS framework](image)

Utilizing this simulation platform, we have demonstrated a modeling approach based on an individual consumer choice model that includes various socioeconomic variables defining sets of static and dynamic input to the model. Particular consideration was given to national data availability and scalability. This modeling and simulation capability allows national simulation of technology penetrations and their impact on climate (CO₂ emission) and electric energy infrastructures. In this spatially explicit model, we developed two novel concepts: a household synthesis model [18] and a simulation of social diffusion of technology adoption using spatial proximity as one of the driving functions. The household synthesis model focused on investigating and developing a dependence-preserving approach in synthesizing household characteristics to support the activity-based traffic demand modeling. For the latter, the simple assumption was made that increasing exposure and awareness of new technology (alternative cars) with and without communication with spatial neighbors (for residents) and colleagues (at work) may provide a positive and a negative impact on potential adaptors. Thus the simulation includes a flexible way to stipulate a distance threshold, which increases or decreases the likelihood of an individual adoption choice. The geographic scalability essentially describes the spatial extent of a particular phenomenon, in this case, the activities of a county’s population, which in turn defines the volume and complexity of the data included in the simulation. Results from the simulation of Knox County, based on a 10% increase in first year PHEV adoption, shows that targeted adoption for families with annual income of $60K and higher could impact 30% more vehicle miles traveled [19].

### 3.3 Large scale monitoring of biomass

Monitoring biomass over large geographic regions for identifying changes is an important for many applications. From deforestation and urbanization, to the recent emphasis on biofuel development for reducing dependency on fossil fuels and reducing carbon emissions from energy production and consumption, the landscape in many sections of the world will change dramatically in coming years. Monitoring crops over large geographic regions is still a challenging problem due to the issue of scaling local/field information to regional scales. Spatiotemporal variability in crop phenological cycles and/or anomalous events, as well as coarse-resolution pixels and mixed pixels, are complicating factors. Such problems are further accentuated in areas where farm plots are relatively small and fragmented across the landscape; such areas may be less amenable to regional remote crop monitoring, as the task of distinguishing between natural and agricultural vegetation may be difficult using traditional methods. Crop monitoring requires reduction of false positives, incorporation of ancillary geospatial data and external events (e.g., floods), scaling of algorithms for large geographic regions, and reduction of human costs by greater automation. In order to understand the changing landscape and complex interactions
between biomass and its environmental variables on a continuous basis, new techniques for monitoring biomass are necessary. Consequently, scalable algorithms for species-level information extraction, from high-resolution (space and time) images, are also required for developing operational systems for biomass monitoring. To address these challenges, ORNL has been developing a novel biomass monitoring framework that consists of two key components: changes detection and change characterization using supervised classification.

Near-real time monitoring vegetation dynamics over large geographic regions is critical for agriculture and bioenergy assurance, and requires high temporal resolution satellite imagery. With the launch of over large geographic region NASA’s Terra satellite in December of 1999, with the MODIS instrument aboard, it is possible to study plant phenology, quantitatively describe net primary productivity (NPP) patterns in time and space, and monitor and map natural resources at regional and global scales. Daily sequence of such satellite derived vegetation data offers temporal continuity but voracity of such data and complexity of scaling local/field information to large geographical extents still poses a geocomputation challenge. MODIS data sets represent a new and improved capability for terrestrial satellite remote sensing aimed at meeting the needs of global change research [20]. Even though several cumulative vegetation indices can be found in the literature, MODIS normalized difference vegetation index (NDVI) temporal profiles are widely used in studying plant phenology. By exploiting recent advances in statistical modeling and machine learning approaches [21], ORNL is designing a generic spatiotemporal data mining framework for monitoring and characterizing changes in croplands using multispectral and temporal remote sensing images and spatial databases.

![Different types of changes in crop biomass](image)

Figure 4: Different types of changes in crop biomass

A novel spatiotemporal data mining framework implemented through hybrid computational architectures have reduced the data analysis time by several orders of magnitude and revealed the potential for such computational approach to be operationalized. This framework provides innovative solutions based on extensions to the state-of-art change detection techniques by taking into account the crop phenology and external (meteorological) events to reduce false-positives (changes). We have developed a novel Gaussian Process (GP) based change detection technique [22] that uses MODIS NDVI time series signals to identify changes. As compared to widely used bi-temporal change detection techniques, our change detection technique continuously monitor the biomass using biweekly MODIS NDVI data, and updates the change map as soon as new NDVI image is inducted into the system. Though our GP based change detection technique showed improved accuracy over other well-known techniques, the computational complexity (time complexity is $O(n^3)$ and memory is $O(n^2)$) of this technique makes it infeasible for large scale biomass monitoring studies. However, our approach incorporates efficient and parallel techniques
using shared and distributed memory models, which make it possible to apply this technique for continuous monitoring of biomass at continental scales. As an example, GP-based change detection technique was able to identify accurately different types of changes as shown in Figure 1. First change indicates a corn (C) field is converted into fallow (F) land, second change indicates a corn field converted into a soybean (S) field, third change indicates corn and soybean rotation is converted into continuous corn, and finally fourth change indicates some kind of damage to the corn fields.

4 Conclusions

Integration, analysis, and visualization of very large volumes of spatiotemporal data are critical and necessary for operational utility across applications of societal significance including distribution and dynamics of population and vegetation on the landscape. At Oak Ridge National Laboratory, research and development based on scalable and data driven geospatial computing have provided encouraging signs of success and progress towards that goal. Our experience indicates that development of novel analytical algorithms for both spatial and spatiotemporal data streams coupled with GPU-based computational approaches can provide processing of very large volumes of satellite based observation data 10 to 200 times faster than previously achieved. Integration of transportation micro-simulations with available crowdsourcing based street network data has allowed local scale modeling and simulation of population and mobility dynamics at global extent. Continuing this impetus towards development novel algorithms and complementary computing architectures hold much promise for enabling the community to address the scientific, technical, and operational challenges of Big Spatial Data.

Acknowledgement

The authors would like to thank a number of U.S. federal agencies for their continued support for the research presented here. Sincere gratitude is due to many of our Geographic Information Science and Technology group colleagues for their collaboration and assistance. This paper has been authored by employees the U.S. Federal Government and of UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. Accordingly, the United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

References


The SIGSPATIAL Special

Section 2: Event Reports

ACM SIGSPATIAL
http://www.sigspatial.org
Highlights from ACM SIGSPATIAL GIS 2014
The 22nd ACM SIGSPATIAL International Conference on
Advances in Geographic Information Systems (Dallas,
Texas, November 4–7, 2014)

Michael Gertz\textsuperscript{1}, John Krumm\textsuperscript{2}, Jagan Sankaranarayanan\textsuperscript{3}

\textsuperscript{1}Heidelberg University gertz@informatik.uni-heidelberg.de
\textsuperscript{2}Microsoft Research jckrumm@microsoft.com
\textsuperscript{3}NEC Labs America jagan@nec-lans.com

(PC Co-Chairs)

1 General

ACM SIGSPATIAL GIS 2014 was held at Dallas, Texas and it was the 22nd gathering of the conference. It is now the seventh time the conference was organized under the auspices of ACM SIGSPATIAL. The conference is the premier event for a variety of researchers, developers, and users who work in related areas to spatial information and GIS. It is an interdisciplinary gathering and provides a forum for original research contributions that cover conceptual, design, and implementation aspects of spatial information systems and GIS.

The attendance for the 2014 conference was 310. The call for papers attracted 216 papers submitted under four categories: research, industry/systems, PhD Showcase, and demo. Specifically, the research and industry/systems categories together attracted 184 submissions, the PhD Showcase category received 3 submissions, and the demonstrations category received 29 submissions. The submissions were reviewed by a program committee of 120 members including 16 meta reviewers. Each paper was reviewed by three reviewers. The resulting program included 39 full research and industry/systems papers as well as 36 research and industry/systems poster papers, and 18 demonstration papers. These numbers demonstrate the continuing success of the ACM SIGSPATIAL GIS and the research field of spatial information and GIS.

The technical program continued the tradition of two and a half days for the main conference with workshops preceding the conference as a separate, single day event. The conference featured the following 12 pre-conference workshops:

- 3rd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data (BigSpatial) 2014
- 3rd ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information (GeoCrowd) 2014
- 1st ACM SIGSPATIAL International Workshop on Privacy in Geographic Information Collection and Analysis (GeoPrivacy) 2014
- 8th Workshop on Geographic Information Retrieval (GIR) 2014
- 3rd ACM SIGSPATIAL International Workshop on the Use of GIS in Public Health (HealthGIS) 2014
• 6th ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness (ISA) 2014
• 7th ACM SIGSPATIAL International Workshop on Computational Transportation Science (IWCTS) 2014
• 5th ACM SIGSPATIAL International Workshop on GeoStreaming (IWGS) 2014
• 7th ACM SIGSPATIAL International Workshop on Location-Based Social Networks (LBSN) 2014
• 2nd ACM SIGSPATIAL International Workshop on Interacting with Maps (MapInteract) 2014
• 3rd ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems (MobiGIS) 2014
• 1st ACM SIGSPATIAL PhD Workshop (SIGSPATIAL PhD) 2014

The 2014 program also featured two outstanding keynotes: *Mapping the World with Street View* by Luc Vincent (Google); and *Interactive Crowd Simulation for Spatial Analysis of Indoor and Outdoor Environments* by Dinesh Manocha (University of North Carolina at Chapel Hill). This year, the conference continued to include a programming contest, the ACM SIGSPATIAL Cup, to its program. The goal of the cup was to encourage innovation and let the community have fun at the same time. It was an exciting event both for the participants and the attendees. The contest was about map generalization, which is the process of reducing the size of the geometry without losing the general shape of a map. The conference also included a business meeting for ACM SIGSPATIAL which was open to all conference attendees and SIGSPATIAL members.

2 Acknowledgements

Finally, we want to thank our sponsor that supported the event. This years conference was generously sponsored by ESRI, Microsoft, Google, Oracle, Facebook, NVIDIA, Yandex and the University of North Texas.
Combining the functionality of mobile devices (smartphones and tablets), wireless communication (Wi-Fi, Bluetooth and 3/4G), and positioning technologies (GPS, Assisted GPS and GLONASS) results in a new era of mobile geographic information systems (GIS) that aim at providing various invaluable services, including location-based services, intelligent transportation systems, logistics management, security and safety, etc. Many mobile GIS applications have been developed to solve challenging real-world problems and improve our quality of life.

MobiGIS 2014 (http://www.mobigis.org) was held in conjunction with the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2014) on November 4, 2014 in Dallas, Texas, USA. It aims at bringing together researchers and practitioners from the GIS community, the mobile computing community, and the data management community. Many current research areas, such as spatio-temporal databases, spatio-temporal data mining, mobile cloud computing, remote sensing, participatory sensing, or social networks, raise research problems that lie at the boundary between these three communities. MobiGIS’s goal is to foster an opportunity for researchers from these three communities to gather and discuss ideas that will shape and influence these emerging GIS-related research areas.

MobiGIS 2014 has received 12 submissions in which 10 research papers were accepted as full research papers and for presentations (25 minutes for each paper). MobiGIS 2014 was a one-day workshop consisting of three sessions: (1) MobiGIS Applications, (2) MobiGIS Services, and (3) MobiGIS Analytics.

We would like to thank the authors for publishing and presenting their papers in MobiGIS 2014, and the program committee members and external reviewers for their professional evaluation and help in the paper review process. We would also like to give very special thanks to our session chairs Prof. Mario Nascimento (University of Alberta, Canada), Prof. Manki Min (South Dakota State University, United States), and Prof. Stanislav Sobolevsky (MIT Senseable City Lab, United States). We hope that the proceedings of MobiGIS 2014 will inspire new research ideas, and that you will enjoy reading them.
Developments in mobile and surveying technologies over the past decade have enabled the collection of individual level geographic information at unprecedented scale. While this large pool of information is extremely valuable to answer scientific questions about human behavior and interaction, privacy intrusion is an imminent risk when detailed individual travel patterns are used for commercial purposes such as customer profiling, or even for political persecution. The GeoPrivacy workshop focused on discussing methods to protect individual’s privacy in geographic information collection and analysis.

GeoPrivacy 2014 (http://stko.geog.ucsb.edu/geoprivacy) was held in conjunction with the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2014) on November 4, 2014 in Dallas, Texas, USA. This workshop touched on an area of geospatial science that affects any researcher working with real-world geodata. With the recent rise in geosocial networking applications as well as advances on location-enabled mobile devices, the topic of geoprivacy has become a major discussion point both in location-specific research as well as everyday life. This workshop offered a unique platform from which to really delve into a dialog on issues related to privacy and credibility within the domain of geoscience and computational geography. The goal of this workshop was to bring together researchers, developers and users of geospatial data to explore methods, techniques, datasets and issues surrounding an area of GIScience that has attracted significant interest among researchers and the public.

The workshop received 10 submissions of which 7 research papers (5 full and 2 short) were accepted for publication in the proceedings and for presentations (30 minutes for each paper).\(^1\) The one-day workshop opened with a thought-provoking keynote from Dr. John Krumm and the remainder of the day was split in to two sessions, Privacy in Location-based Services and Privacy Protection. The GeoPrivacy workshop concluded with a group discussion session which involved a round-table dialog on communal goals, ongoing projects and future directions for research.

We would sincerely like to thank the authors for publishing and presenting their work at GeoPrivacy 2014, the keynote speaker and the program committee members and external reviewers for their thoughtful evaluation and help in the paper review process. We hope that readers of the workshop proceedings will find it interesting and it will motivate continued discussion on the future of geoprivacy-based research.

\(^1\)The proceedings are accessible at http://dl.acm.org/citation.cfm?id=2675682
HealthGIS 2014 Workshop Report
The Third ACM SIGSPATIAL International Workshop on the Use of GIS in Public Health
Dallas, Texas, USA - November 4, 2014

Daniel Goldberg\textsuperscript{1}, Ori Gudes\textsuperscript{2}, Yaron Kanza\textsuperscript{3}
\textsuperscript{1}Texas A&M University, USA, daniel.goldberg@tamu.edu
\textsuperscript{2}Department of Spatial Sciences, Curtin University, Australia, ori.gudes@curtin.edu.au
\textsuperscript{3}Jacobs Institute, Cornell Tech, USA, kanza@jacobs.cornell.edu
(Workshop Co-Chairs)

The Third ACM SIGSPATIAL International Workshop on the Use of GIS in Public Health was held on November 4, 2014 in Dallas, Texas, in conjunction with the 22st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. Following two previous successful HealthGIS workshops \cite{1, 2}, it has brought together researchers whose research is in the intersection of geospatial data management and public health. This workshop provided a forum for researchers and practitioners to share new ideas and techniques for Health-related GIS applications. Original research related to all aspects of GIS usages and applications in medical and in healthcare systems was invited; especially papers based on real-world experience were encouraged. The program committee consisted of a diverse set of members from academia, industry, and government research laboratories, each with expertise in different areas of HealthGIS: development, applications, and public health research. The workshop has provided a platform for an interdisciplinary dialog that is reflected in the program of the workshop.

A total of 13 paper submissions were received out of which 9 papers were accepted—two position papers and 7 full papers. These papers were broken into three sessions:

1. Guidance and Assistance Applications;

2. Mapping and Tracking Diseases;


The first session focused on applications for guiding people and assisting patients. Topics covered in this session included the use of games in physical therapy, detection of freezing of gait by analysis of movement, to assist Parkinson’s Disease patients, and applications to encourage users to perform physical activities in their daily life. The second session focused on topics related to tracking of diseases. This session included a study of the track of diseases based on news articles, visualization of hepatitis outbreaks and probabilistic models for mapping diseases. The third session focused on analysis and monitoring of health data. Papers in this session included a spatial analysis of substance abuse treatment for low-income and minority households, and the use of address history in health data, focusing on the difficulties caused by incomplete and inaccurate addresses when investigating issues related to public health. Between the second research session and the third research session there was a session that included two invited talks.
The first invited talk was about “Privacy in Location-Based Health Services” and it was given by Lars Kulik from the University of Melbourne, Australia. The second invited talk was about “Disease, Vulnerability and GIS” and it was given by Joseph Oppong from the University of North Texas.

The workshop organizers sought to bring together a diverse set of participants representing voices from GIS, Computing, and Public Health. By all measures, the Third ACM SIGSPATIAL International Workshop on the Use of GIS in Public Health (HealthGIS 2014) was a success. The papers received and presented, and the participants who attended represented a diverse group spanning practitioners to researchers, academics to professionals, with interests and expertise equally as broad. There were about 20 participants in the different sessions. The panel session and the concluding discussion included lively and constructive discussion from many of these perspectives. This venue offered an opportunity for research and ideas to span traditional boundaries of disciplines. All participants capitalized on the opportunity to meet and develop relationships with those from other disciplines working on related research or application problems. This served the workshop continuing goal of building a community of HealthGIS researchers and practitioners within the Spatial Computing, GIS, and Public Health communities, in order to develop creative solutions drawing from the best research ideas in each of these disciplines to help relieve the burden of disease worldwide and to improve public health services.

We would like to thank the authors for publishing and presenting their papers in HealthGIS 2014, the program committee members and the external reviewers for their professional evaluation of the submitted paper, the members of the panel and the speakers of the invited talks—Lars Kulik and Joseph Oppong. We also want to thank the participants who made the workshop as lively as it was and the session chairs. Last, but not least, we would like to thank the ACM SIGSPATIAL 2014 Workshops chairs, Mohamed Mokbel and Egemen Tanin for their great organization of the workshops, and thank Hanan Samet and Deborah Estrin for their advice, guidance and support as the advising committee.

As for the future prospect of this workshop, we hope that it will span traditional boundaries of the involved communities by creating synergies between researchers within the spatial science domain and health practitioners, and by promoting the new research domain of the use of GIS in public health.

References


A large proportion of information on the web and in digital libraries is referenced in some way to geographical locations, but conventional geographical information systems technology is not oriented to managing and analyzing the largely unstructured, often textual form of much of the information. The subject of Geographical Information Retrieval (GIR) is concerned with providing methods to access this relatively unstructured geographical information embedded in digital documents on the web and elsewhere. GIR uses and builds upon methods both from geographical information systems (GIS) technology, which is designed to access structured geo-spatial data based on digital maps, and from the field of Information Retrieval (IR) in which the emphasis is on access to text documents. There are many research challenges in developing effective GIR systems, relating for example to detection and disambiguation of references to geographic information in text, spatial indexing of documents and their content, development of spatially aware search engines, and visualization of geo-information. The Workshop on Geographic Information Retrieval provides a forum to discuss these issues and to present new research results.

The GIR workshop held on 4th November 2014 at the ACM SIGSPATIAL conference in Dallas, Texas, was the eighth in a series that started in 2004 and has been held previously in combination with the SIGIR and CIKM conferences. The workshop has continued to attract a stimulating mix of researchers and practitioners from a variety of academic disciplines and industrial backgrounds.

At GIR’14 there were 11 presentations, all of which were full papers. The workshop was organized around four sessions that aligned with the common themes of the submitted papers. The first session, on search and retrieval methods, included papers on the extraction of itineraries from spreadsheets and tables in web documents; extending spatial-keyword (spatio-textual) indexing to the temporal dimension; and indirect location recommendation based on the content of a user’s personal social network. The second session on resources for GIR included two papers on geo-spatial corpus production, for geo-parsing text in micro-blogs (Twitter) and for spatial relations (such as near, close to and next to) between geo-located named entities; and a paper on building a gazetteer from linked data sources (Geonames and DBpedia). The third session was concerned with the subject of toponym detection and resolution, focusing on the use of different types of rhetoric and tropes to characterize the different ways in which toponyms are used; the exploitation of Wikipedia to provide contexts for helping to disambiguate toponyms; and the use of maps to help users to disambiguate toponyms. In the final session, on analyzing events in social media, one of the papers analysed the different quality of the accounts of witnesses of events as evidenced from micro-blog content; while the other presented a method for determining the type of an event based on a machine learning approach that exploits the tags of nearby Flickr photos.
The ACM Special Interest Group on Spatial Information (SIGSPATIAL) addresses issues related to the acquisition, management, and processing of spatially-related information with a focus on algorithmic, geometric, and visual considerations. The scope includes, but is not limited to, geographic information systems (GIS).

The Association for Computing Machinery (ACM) is an educational and scientific computing society which works to advance computing as a science and a profession. Benefits include subscriptions to Communications of the ACM, MemberNet, TechNews and CareerNews, full and unlimited access to online courses and books, discounts on conferences and the option to subscribe to the ACM Digital Library.

- SIGSPATIAL (ACM Member) ................................................................. $ 15
- SIGSPATIAL (ACM Student Member & Non-ACM Student Member) ........................................ $ 6
- SIGSPATIAL (Non-ACM Member) ........................................................................ $ 15
- ACM Professional Membership ($99) & SIGSPATIAL ($15) ........................................ $114
- ACM Professional Membership ($99) & SIGSPATIAL ($15) & ACM Digital Library ($99) ........................................ $213
- ACM Student Membership ($19) & SIGSPATIAL ($6) .................................................. $ 25

Mailing List Restriction
ACM occasionally makes its mailing list available to computer-related organizations, educational institutions and sister societies. All email addresses remain strictly confidential. Check one of the following if you wish to restrict the use of your name:
- ACM announcements only
- ACM and other sister society announcements
- ACM subscription and renewal notices only

Questions? Contact:
ACM Headquarters
2 Penn Plaza, Suite 701
New York, NY 10121-0701
voice: 212-626-0500
fax: 212-944-1318
email: acmhelp@acm.org

Remit to:
ACM
General Post Office
P.O. Box 30777
New York, NY 10087-0777

www.acm.org/joinsigs  Advancing Computing as a Science & Profession