4SA: OPTIMIZING SPACE FILLING CURVE BASED GRID CELL INDEXING TO SCALABLY MANAGE REMOTELY SENSED IMAGES IN KEY-VALUE DATABASES

C.N. Lokugam Hewage[1,*], A.V. Vo[1], M. Bertolotto[1], N-A. Le-Khac[1], D. Laefer[2,3]

[1]School of Computer Science, University College Dublin, Ireland - chamin.lokugamhewage@ucdconnect.ie,
(anhvu.vo, michela.bertolotto,an.lekhac)@ucd.ie
[2]Center for Urban Science and Progress, New York University, USA - debra.laefer@nyu.edu
[3]Department of Civil and Urban Engineering, New York University, USA

Commission IV, WG IV/9

KEY WORDS: remotely sensed images, key-value databases, space filling curves, grid indexing, scalability

ABSTRACT:

State-of-the-art remote sensing image management systems adopt scalable databases and employ sophisticated indexing techniques to perform window and containment queries. Many rely on space-filling curve (SFC) based index techniques designed for key-value databases and are predominantly employable for images that are iso-oriented. Critically, these indexes do not consider the high degree of overlap among images that exists in many data sets and the affiliated storage requirements. Specifically, employing an SFC-based grid cell index approach in consort with ground footprint coverage of the images requires storage of a unique image object identification (IOI) for each image in every grid cell where overlap occurs. Such an approach adversely affects both storage and query response times. In response, this paper presents an optimization technique for an SFC-based grid cell space indexing. The optimization is specifically designed for window and containment queries where the region of interest overlaps with at least a 2 x 2 grid of cells. The technique is based on four cell removal steps, thus called "four step algorithm" (4SA). Each step employs a unique spatial configuration to check for continuous spatial extent. If present, the IOI of the target cell is omitted from further consideration. Analysis and experiments on real world and synthetic image data demonstrated that 4SA improved storage demands by 41.3% - 47.8%. Furthermore, in the performed querying experiments, only 42% of IOI elements needed to be processed, thus yielding a 58% productivity gain. The reduction of IOI elements in querying also impacted the CPU execution time (3.0% - 5.2%). The 4SA also demonstrated data scalability and concurrent user scalability in querying large regions by completing the index searching and concurrent user scalability 1.86% - 3.35% faster than when 4SA was not applied.

1. INTRODUCTION

With the advent of high resolution cameras and more affordable modes of aerial and space-borne imagery, there has been an acute growth in a wide variety of remotely sensed imagery data (RSID) (Wang et al., 2019). Such high resolution RSID provide a rich source of information for a multitude of geo-spatial applications such as spatial data lakes for smart cities (Kafando et al., 2020). Consequently, increased research efforts are being geared to develop efficient RSID storage and querying systems. Ensuring the ability to cope with increased RSID volumes, while maintaining commensurate storage and querying performance, are key objectives in these efforts. In other words, providing scalable, efficient storage for data growth and retrieval by large number of users and for RSID data sets of ever increasing size is pivotal for the future of scalable RSID management.

In confronting scalability challenges in RSID, the investigation of novel spatial indexes has been a key driving force. Typically, to cope with large data sets and voluminous traffic loads, such investigations have been performed atop state-of-the-art databases that are themselves inherently scalable through their distribution of loads across multiple machines. As a result, horizontally-scaled, shared-nothing architecture based RSID storage and retrieval solutions are gaining traction in contemporary RSID data management. These indexing techniques are mainly geared towards to support window queriesn and/or containment queries -i.e. retrieving images or image tiles that are at least partially overlapping with the query region of interest.

Prominent examples include, GeoMesa (Hughes et al., 2015), RASDAMAN (Baumann et al., 1997), TileDB (Papadopoulos et al., 2016), RSIMS (Zhou et al., 2021), and Wang et al. (2015). Their databases RASDAMAN, TileDB, and RSIMS employ horizontally scaled, object relational databases. GeoMesa and systems from Wang et al. (2015) and Jing and Tian (2018) are built atop Key-Value (KV) databases - a class of NoSQL database that is inherently scalable for high data growth demands and traffic volume requirements. Both RASDAMAN and TileDB implement R+-tree and R-tree indexes by organizing the minimum bounding rectangles (MBRs) of RSID. RSIMS and the aforementioned KV database oriented systems implement Space Filling Curve (SFC) based indexes. When organizing two-dimensional (2D) MBRs, the R-tree and R+-tree enable overlapping of regions. Thus, space partitioning is not uniform.

In contrast, when adopting SFC based indexing, typically the space is divided into different levels of grids where every grid cell in each level has a uniform spatial resolution. Due to this, current RSID systems that employ SFC-based grid cell, space indexing organize images or image tiles into a specific grid cell in a particular grid level according to the spatial coverage of the image. More specifically, when adopting SFC-based grid cell space indexing, the images tend to be indexed in one of two ways. The first approach is to divide images into axis-aligned or iso-oriented tiles and perform the indexing on the image tiles.

* Corresponding author

The second approach involves applying an SFC-based grid cell space index onto precisely formatted iso-oriented images.

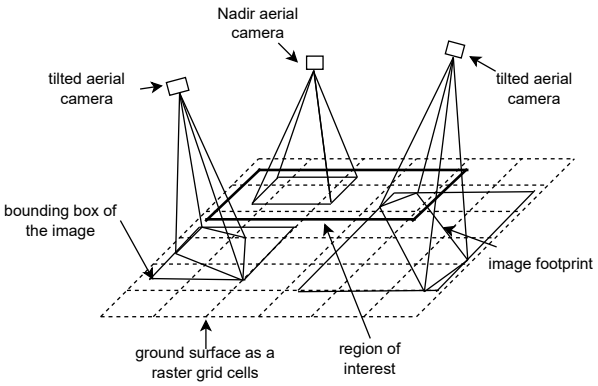

Figure 1. ground footprint of aerial images

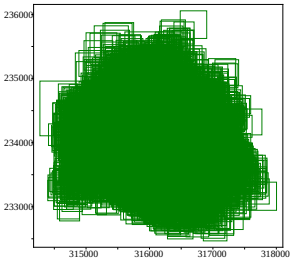

Figure 2. High degree of overlap among images

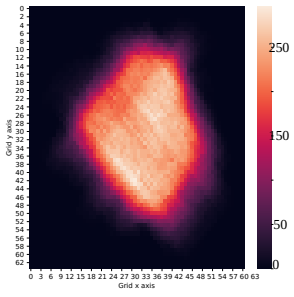

Figure 3. Heat-map of image overlapping

However, the originally captured raw RSID in image mapping projects are often not iso-oriented and/or not precisely formatted to fit into grid cells. Figure 1 demonstrates the *near similar*[1] spatial area coverage of a set of raw RSID that were captured through different airborne cameras such as *nadir* cameras and *tilted* cameras. For example, the laser scanning and imagery mapping exercise performed in the city of Dublin, Ireland in a 2015 were mainly based on a set of *nadir* and *tilted* cameras (Laefer et al., 2017). Due to the angle of tilted cameras, the ground footprint or the spatial coverage of the RSID capture are not iso-oriented. Similarly, although *nadir* cameras are mounted vertically to the earth, due to small rotations and tilts in the mapping platforms during real data capture, the resulting images are neither strictly iso-oriented nor precisely sized into a grid cell arrangement. Thus, the direct application of SFC-based grid cell space indexing to raw RSID data is challenging.

Images with a high degree of overlap is another recognized characteristic among raw RSID. Figure 2 demonstrates the overlap among the bounding boxes of the RSID captured in Dublin in 2015 image mapping project. Each image bounding box is on average about 400 meters (m) in its height and width. Figure 3 corresponds to the heat map of the RSID overlapping with respect to a 64 x 64 grid where each unit cell has a height and width of 60 m. Figure 3 demonstrates that in real world RSID mappings, the number of RSID that can overlap with a given geographic location can amount to hundreds of images. A potential solution to address the high degree of image overlap can be designed by employing SFCs with much larger, coarse-grid cells (e.g. 500 m x 500 m). However, this strategy is less than ideal for representation of much smaller sized RSID. Thus, adopting an SFC-based cell space indexing poses specific challenges when handling raw RSID collected at different scales and distinctive resolutions. These challenges are applicable across both aerial and space-borne raw RSID management.

Importantly SFC-based indexing can be tailored to spatially organize raw RSID. Specifically, as SFCs decompose the space into a series of grid cells, the cells that comprise each RSID can be used to store the image object id (IOI) of each RSID (Gaede and Günther, 1998; Samet, 2006). Since multiple RSID can overlap with a single grid cell (cf. Figure 3), this would necessitate storing multiple IOIs in that individual cell. Once the IOIs for each grid cell are determined, the cell can be stored in a highly scalable system such as a KV database. Adoption of a KV database would make the SFC values associated with each grid cell be the keys in a key-value arrangement. The list of IOIs contained in each cell then becomes the values for each key in the key-value pairs.

While the described solution is scalable, it demands the storage of the IOI of each RSID in every cell for all RSIDs for which there is overlap. Unless the size of the unit grid cell is quite coarse, the storage of the IOIs across every cell covering each RSID would cause excessive storage consumption (Böhm et al., 1999). This challenges the storage viability requirements in adopting SFC-based cell space indexing approaches. Additionally, storage of every overlapped IOIs would also lead to processing an excessive number of IOIs in the respective window and containment query algorithms. Thus, these query response times would then be negatively impacted, as well.

In response to adopting SFC index techniques and impediments arising from excess storage of in SFC-based grid cell indexing, this work proposes "four step algorithm (4SA)". The 4SA is an optimization technique designed to apply to SFC-based grid cell indexing for raw RSID management in KV databases. The goal of 4SA is to maintain the bare minimum number of IOIs required for each RSID. As the name implies, 4SA is based on the execution of four steps. The adoption of 4SA is specifically designed towards window and containment queries when the query region overlaps a minimum 2 x 2 grid of cells. The experiments and critical analysis of this paper are based on both real world aerial data and synthetically generated data sets. The results show 4SA scalability, in terms of storing large data sizes and querying significant spatial extents with multiple concurrent users. The contributions of this work are:

- Introduction of the 4SA- an algorithmic optimization for SFC based grid cell space indexing for RSID management within KV databases through the reduction of IOI storage for RSID.
- Demonstration of the effectiveness of 4SA in improving storage performance, and intuitively, a qualitative demonstration of minimization of write/ update costs of indexes.
- Demonstration of reduction of the processing of IOI elements and reduction CPU time for window and containment queries as a result of applying 4SA.
- Demonstration of data scalability and concurrent user scalability through the adoption of 4SA in querying.
- Statistical and numerical comparisons of incorporating the 4SA for straightforward SFC-based grid indexing adoption and region querying.

[1] In urban areas the footprints are not precise quadrangles, the footprints are disturbed by relief and occlusion of cameras. Figure 1 is provided for the purpose conveying the idea that raw images are not iso-oriented.

2. RELATED WORK

In this section, prominent state-of-the-art, database-oriented RSID management solutions are presented. The indexing techniques adopted, the scalable databases used, and their limitations are also presented in brief.

RASDAMAN (Baumann et al., 1997), which uses a multidimensional, array data model for processing spatial raster data, is one of the leading solutions for storage and retrieval of RSID. The main focus of RASDAMAN is window and range queries. RSID in RASDAMAN are stored as binary large objects inside a PostgreSQL database. In its indexing, RASDAMAN uses built-in index structures such as R+-tree and GiST indexes. TileDB (Papadopoulos et al., 2016), another array data model based RSID storage and retrieval system, also employs an R-tree index to support window queries. Since the R-tree and the R+-tree allow overlapping of indexed spatial extents, the raw RSID can be included in multiple overlapping spatial regions. Thus, a high degree of overlapping can be solved, to some extent. However, HBase KV, database-oriented, spatial systems that implement an R-tree maintain it as an in-memory data structure (Huang et al., 2014). When there are a tremendous amount of overlapping images, such an approach is unfeasible. Furthermore, use of in-memory data structures in HBase settings will impede scalability issues when dealing with high volumes of RSID indexes in the main-memory.

The RSIMS (Zhou et al., 2021) which is built atop a PostgreSQL database cluster and Ceph distributed object storage system, also supports storage and retrieval of RSID. In its indexing layer, RSIMS employs a distributed multi-level, Hilbert index. Specifically, when indexing images or image tiles, first it determines the level at which the spatial index needs to be calculated. The level is based on the spatial coverage or the spatial area of the respective image. Once determined, the corresponding Hilbert index for the image is calculated through the latitude and longitude of the geometric centre of the image. When considering raw RSID, obtaining precise geometric positions is non-obvious due to the absence of iso-orientation and/or formatting. Thus, neither RSIMS as a system, nor as an index strategy, is well suited to be adopted for management of raw RSID.

Wang et al. (2015), Jing and Tian (2018), and GeoMesa are three state-of-the-art RSID systems built atop KV databases. Wang et al. (2015) is based on division of the earth's surface into different grids and indexing of each grid cell's latitude and longitude on GeoSOT, a global discrete grid system. Jing and Tian (2018) implements Hilbert like grid cell space indexing. These two strategies organize RSID with different spatial coverage into cells of specific resolutions in various grid levels. This technique avoids IOI duplication of RSID. However, this strategy can only be implemented for RSIDs that are precisely fit into grid cells and images that are iso-oriented. Thus, the two systems and their index strategies cannot be adopted directly for raw RSID storage and retrieval.

GeoMesa, built atop the Accumulo KV database, also supports window queries. GeoMesa employs XZ-SFC (Böhm et al., 1999) for indexing of image tiles. Based on the image resolution, these image tiles are organized into multiple levels to form an image pyramid. When querying, GeoMesa returns image tiles from appropriate levels of the pyramid. The XZ-curve uses the concept of *extended region*. The concept of *extended region* applies when the spatial object overlaps with multiple grid cells. In such scenarios, to avoid object id duplication across multiple cells, a much larger region is formed. This is done by increasing the height and width of the bottom left cell by a factor of 2 upwards and to the right. The concept of *extended region* works for objects with spatial extents in general. However, in the original work of XZ-SFC, the authors did not discuss how this concept could be applied when non-point objects have a high degree of overlap among them. Therefore, in real world aerial image mappings, where the image objects overlapping tends to have a high degree, the adoption of an XZ-SFC-curve would form a single grid cell. Consequently, all RSID would reside in a single, large grid cell, which is untenable. Therefore, GeoMesa and its XZ-SFC-index strategy as currently structured cannot be adopted efficiently for raw RSID management.

In addition, as previously noted, SFC-based image indexing organizes images into grids of multiple levels. This can be considered as a strategy to avoid IOIs across multiple grid cells, if one grid is adopted. However, this technique demands prior knowledge of RSIDs with respect to the ground footprint of each RSID to correctly assign it to the appropriate grid. The ground footprints of RSID can have a wide spectrum of values. Thus, pragmatically organizing all RSID into an immutably, pre-defined grid is unfeasible.

To address the research gaps in managing raw RSID, while ensuring scalability and considering the unique characteristics inherent to raw RSID, investigation of new research avenues is pivotal. As described in Section 1, a solution can be devised by integrating an SFC-based grid cell space indexing, while adopting a scalable KV database. However,it demands excessive IOI storage. In response to that limitation, this work proposed 4SA optimization. Section 3 describes the rationale of 4SA.

3. METHODOLOGY

The objective of 4SA is to store the bare minimum number of IOIs when adopting an SFC-based cell indexing for RSID management in KV databases. The 4SA is also geared toward window and containment queries where the query regions overlap with at least a 2 x 2 set of grid cells. Maintaining the bare minimum number of IOIs is determined through the spatial coverage or the footprint of each RSID. More specifically, when an image overlaps with multiple grid cells, each step in of the 4SA attempts to determine if an IOI in a particular cell satisfies a certain spatial configuration (SC) type that characterizes the overlap. These SC types in the 4SA are based on the adjacency of relevant cells in terms of cells that are directly next to or diagonal to the cell under investigation. In the 4SA, there are four SC adjacency shapes. Each is named based on the shape of its adjacency grid (cf. Figure 4): (i) type "+" (i.e. type "plus"), (ii) type "X", (iii) type "Ts" (i.e. different rotations of the letter "T"), (iv) type "ls" (i.e. different rotations of the letter "l"). If an IOI in a specific cell satisfies a SC type in one of the steps, then the 4SA omits storage of the IOI in the respective cell, and no further consideration is given to that cell.

The rationale for selecting these four SC types was motivated by minimization of IOI storage through two kinds of grid cells: (i) those internal to the image's footprint and (ii) those that reside along the footprint's border. Typically, the number of cells that reside within an image's footprint is much higher than the number of cells that reside along its footprint (cf. Figure 5). Thus, elimination of IOI storage in cells internal to the image's footprint is prioritized by incorporating the concepts in *4-connected pixels* and *8-connected pixels* rationales.

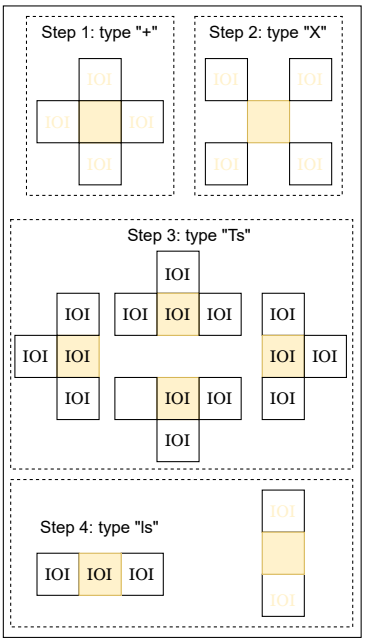

Figure 4. Spatial configuration types in each step of 4SA

In the rationale of *4-connected pixels*, the center pixel is the pixel of interest and touches the borders of four other pixels (2 vertical and 2 horizontal). The rationale of *8-connected pixels* additionally includes the four pixels that touch the corners in the two diagonal directions. As the prime focus in both the 4-connected pixels and the 8-connected pixels is the direct association with the internal cells, two SC types were selected as the initial pair of 4SA steps to avoid IOI duplication of inner cells. These are type "+" and type "X". Type "+" is synonymous with *4-connected pixels*. However, in defining the SC for type "X", only the cells that touch the corners are considered.

While, the output of steps 1 and 2 increase in the ratio of cells along the image's ground footprint to those within the footprint, many interior cells remain. Thus, a further two SC types are defined and implemented in steps 3 and 4. The objective of the third SC type is to eliminate the IOI storage in cells that reside both along the image's footprint (i.e. the boundary) and within its interior. Thus, the SC type "Ts" was devised. Finally, to minimize the IOI storage of grid cells that reside along the image's footprint, the SC type "ls" was introduced as the shape for step 4.

When storing IOI for an image, 4SA retains both a plethora of cells that reside along an image's footprint and a limited number that are internal to the image's footprint. This design decision was conceived as an alternative to total elimination of all internal cells. The rational was to help ensure that much smaller queries (e.g. a 2 x 2 cell query) would not falsely omit larger, image query regions that do not cross the border (i.e. edge) of the larger image's boundary.

3.1 Four Step Algorithm (4SA)

Algorithm 1 takes a collection of RSID (i.e. SET_{RSID}) as input for its execution (Line 1). Each image i_{img} in the SET_{RSID} overlaps with a set of grid cells (i.e. $SET_{i_{gc}}$). As such, each grid cell i_{gc} in the $SET_{i_{gc}}$ contains the IOI of the respective i_{img}. To successfully apply the 4SA, specific knowledge about the i_{gc} of each i_{img} is essential. Therefore, the calculation of $SET_{i_{gc}}$ for each i_{img} is required a priori to the execution of the four steps and is achieved by utilizing the ground footprint of each i_{img} and coordinate values of the intended grid cells.

The output of 4SA contains a subset of grid cell ids for each image and can be defined as SET_{RSID}. The four steps are executed sequentially. The execution of the 4SA algorithm starts by looping through each RSID i_{img} in the SET_{RSID} (Line 2). For each i_{img}, the algorithm extracts the set of grid cells where the IOI of the respective i_{img} is stored (Line 3). This set represents the full collection of grid cells that the IOI of the i_{img} stores. The execution of the four steps on the collection of the i_{gc} of the respective i_{img} are subsequently performed.

The output of each step becomes the input grid cell set (i.e. $SET_{i_{gc}}$) for the next step. Thus, each sequential step in the 4SA analyzes a reduced set of data. Specifically the $SET_{i_{gc}}$ is analyzed based on each i_{gc} in the $SET_{i_{gc}}$. This is the function of the inner for loops of each step in the Algorithm 1. The method of analyzing each i_{gc} within each loop is based on the SC types (i.e. step 1: sc_+, step 2: sc_X, etc.). If any of the SC types exist (i.e. $\exists$) within the currently processing i_{gc}, the i_{gc} will be removed from its $SET_{i_{gc}}$ for the considered i_{img} (i.e. $SET_{i_{gc}}.remove(i_{gc})$). The process is shown in Figure 5.

Algorithm 1: Four Step Algorithm (4SA)

Input: SET_{RSID} ▷ RSIDs with actual spatial coverage
Output: SET_{RSID} ▷ a filtered sub set of SET_{RSID}

1 **Function** `FourSA(`SET_{RSID}`):`
2 **foreach** $i_{img} \in SET_{RSID}$ **do**
3 $SET_{i_{gc}} \leftarrow i_{img}$
 ▷ Step 1: check for type "+" SCs
 foreach $i_{gc} \in SET_{i_{gc}}$ **do**
 if $sc\_+ \ \exists \ i_{gc}$ **then**
 $SET_{i_{gc}}.remove(i_{gc})$
 end
 end
 ▷ Step 2: check for type "X" SCs
 foreach $i_{gc} \in SET_{i_{gc}}$ **do**
 if $sc\_X \ \exists \ i_{gc}$ **then**
 $SET_{i_{gc}}.remove(i_{gc})$
 end
 end
 ▷ Step 3: check for type "Ts" SCs
 foreach $i_{gc} \in SET_{i_{gc}}$ **do**
 if $sc\_Ts \ \exists \ i_{gc}$ **then**
 $SET_{i_{gc}}.remove(i_{gc})$
 end
 end
 ▷ Step 4: check for type "ls" SCs
 foreach $i_{gc} \in SET_{i_{gc}}$ **do**
 if $sc\_ls \ \exists \ i_{gc}$ **then**
 $SET_{i_{gc}}.remove(i_{gc})$
 end
 end
4 **end**
5 **return** SET_{RSID} ▷ final filtered sub set of SET_{RSID}
6 **End Function**

3.2 Example of applying 4SA

Figure 5 presents the steps of 4SA, while demonstrating how the described SC types can be used to avoid IOI storage in grid cells when adopting SFC oriented grid cell spaces indexing. This example is based on a real aerial image of Dublin.

As shown in Figure 5a, the footprint of the aerial image either fully or partially overlaps with a total of 36 grid cells. Thus,

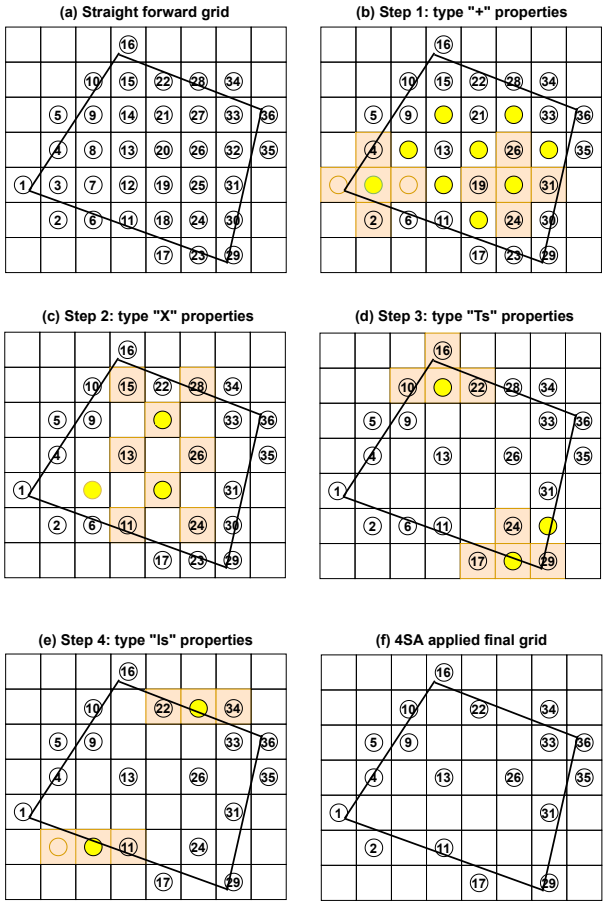

Figure 5. Application of 4SA on real-world RSID

adopting an SFC-based grid cell space translates into indexing that will store IOIs in 36 cells. Sub-figure 5b checks type "+" SC types. The 9 grid cells that satisfy type "+" SCs are marked in yellow (3, 8, 12, 14, 18, 20, 25, 27, and 32); for demonstration purposes, the cells that contribute to the inclusion of cells 3 and 25 are colored in light orange). In this example, step 1, reduced the candidate cells to 27, thereby achieving a 25% reduction. The remaining cells become the input for step 2.

The influence of applying type "X" in step 2 is presented in subfigure 5c. The three grid cells that satisfy the SC type "X" are 7, 19, and 21 and are marked in yellow. The cells that contribute to qualification of cells 19 and 21 are coloured in light orange. Step 2 reduces the candidate cells by 3 (a further 8%), leaving 24 cells as the input for step 3. Application of type "Ts" SCs - i.e. different rotations of letter "T" are presented in sub-figure 5d. Three grid cells (15, 23 and 30) satisfied the criterion, thereby reducing the candidate pool by a further 8% and leaving 21 candidate cells as step 4 input. In step 4, cells 6 and 28 in sub-figure 5e satisfy the SC type "ls", thereby reducing the candidate pool an additional 5.6% for a total reduction of 46.6%. The final 19 cells demonstrate the minimum number of cells whose IOIs that need to be stored for the RSID.

4. EVALUATION

This section presents the impact of applying 4SA to SFC-based grid cell space indexing. An extensive scalability study and statistical analysis of applying 4SA against non-4SA applied scenarios are also presented.

4.1 Study data

All experiments were grounded in a real world aerial image data set of the city centre of Dublin, Ireland. These include 8,438 aerial images covering an approximate region of 2km^2. They were captured during an airborne laser scanning and imagery mapping exercise in 2015. The imagery primarily consists of two classes: (i) oblique images captured from two separate tilted cameras, (ii) Red-Green-Blue (RGB) [i.e. true colored] and colored infrared images captured via a nadir camera (Laefer et al., 2017). As the main focus of the experiments was to demonstrate the impact of 4SA on scalable management of RSID, this 8,438 image data set was also used as the basis to generate a series of much larger synthetic data sets. The largest synthetic image data set consisted of a total of 303,768 RSID metadata records. The area coverage of the synthetically generated data set accounted for 576km^2 (i.e. 24 km x 24 km region).

4.2 Experimental set-up

The synthetic data set generation and application of 4SA in generating the filtered grid based on the Algorithm 1 mentioned in Section 3 were performed locally. Scalability experiments were conducted on the Peel cluster at New York University (NYU). The Peel cluster includes 2 login nodes and 18 high-end compute nodes. The hardware specifications of the nodes are as follows: 2x24 - core Intel ® Xeon ® Platinum 8268 CPU @ 2.90 GHz, RAM - 384 GB, disk storage - 8 HDD disks 8 TB each. Apache Hadoop (version 3.0.0) and HBase (version 2.1.0-cdh6.3.4) are deployed in the Peel cluster for big data management tasks.

4.3 Statistical analysis of each step in 4SA

Statistical evaluations on both the real world and synthetic data sets were performed to further deepen understanding of 4SA on reduction of IOI storage in grid cells. For the real world data set, a 64 x 64 grid with a cell resolution of 60 m x 60 m was adopted. For the synthetic data set, a 320 x 320 grid with a cell resolution of 75 m x 75 m was generated. While executing the 4SA on the actual ground footprints of both the real and synthetic data sets, a quartet of additional counter variables for each step were used to better understand the impact of each step on the overall IOI storage reduction. The final values of each counter variable in all steps were evaluated against the initial total IOI count, with respect to the entire images in the two data sets. The results obtained are presented in Table 1.

As shown Table 1, in both data sets, the total reductions in IOIs were approximately 40% (i.e. 41.8% for real data set and 38.8% for synthetic data set). Notably, the contribution of step 1 exceeded 20% in both cases. Thus, step 1 in the 4SA process dominated. Intuitively, this can be attributed to step 1 being the first executing step and subsequent steps only receiving a subset of cells as input. Furthermore, intuitively, the ground footprints of RSID occupy more inner cells than the cells that constitute the border of the ground footprints, and step 1 mainly impacts cells that reside inside ground footprint. This can be recognized as another reason for using the "+" type for step 1. Although the contribution of the other three steps is comparatively low, cumulatively they accounted for nearly half of the overall set of cells. Another important observation is the similarity of percentage drops by step for every step in both data sets, with more than half of the reduction occurring in step 1. This may have been influenced by the similar grid resolutions for the two data sets (i.e. 60m x 60m and 75m x 75m). To arrive at robust

conclusions, more methodical tests on RSID that have different spatial coverage are planned to be performed in future.

Table 1. Impact of each step on overall reduction of IOI storage

Step	Actual data set	Total drop as a %	Synthetic data set	Total drop as a %
Before 4SA	280,847	-	6,988,360	-
1	209,236	25.5	5,447,696	22.0
2	190,336	6.7	5,087,116	5.2
3	180,107	3.6	4,753,768	4.8
4	163,530	5.9	4,299,700	6.5
After 4SA	163,530	41.8	4,299,700	38.8

4.4 Impact on scalable storage

To evaluate scalability with respect to growth of RSID index storage requirements, tests were performed by measuring disk usage requirements of three different scenarios with the use of synthetic data sets and a 320 x 320 grid. For each scenario, the data were compared when processed with and without the application of 4SA. Prior to measuring the disk usage of each, the IOI storage requirement for each image in a 320x320 grid was determined. The 6 data sets (i.e. 3 scenarios x 2 data sets) were ingested into separate HBase tables from which their disk usage was determined. Figure 6 presents the comparison of storage performance.

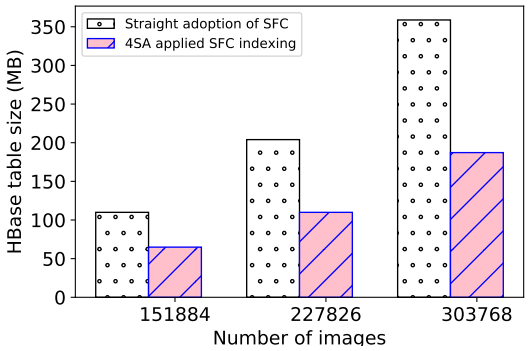

Figure 6. Storage requirements for SFC based indexing with and without 4SA

The impact of adopting 4SA is depicted in Figure 6. In all three scenarios, the straightforward SFC-based grid cell space index demands were compared to the 4SA optimized solution. The improvements in adopting 4SA were 41.3% for 151,884 images, 46.6% for 227,826 images, and 47.8% for 303,768 images. This demonstrates that the 4SA optimization on SFC-based cell space indexing technique scaled better than the straight forward adoption of the SFC indexing baseline.

Storage improvements in Figure 6 demonstrate that the application of 4SA over a large data set required to write considerably fewer IOIs to the database. Therefore, requiring less storage. Intuitively, that means, the application of 4SA results in fewer database writes when organizing each RSID in respective grid cells, as opposed to non-4SA applied scenarios where the IOIs must write to all the grid cells. Similarly, if updating (or modifying) an IOI for a specific RSID is required, the application of 4SA minimized the number of grid cells that must be updated/ modified. Consequently, as a qualitative assessment, 4SA also enables the minimization of write/update costs associated to SFC based cell indexing. For example, for the real world image in Figure 5, qualitatively, it can be stated that application of 4SA only required to update its IOI from 19 cells instead 36 cells. Thus updating cost for the image is expected to reduce by 46.6% for the example image.

4.5 4SA's impact on scalable querying of RSID

Querying scalability was also tested in the HBase cluster. For this, the largest synthetically generated data set containing 303,768 RSID meta data records was used. The impact of 4SA was evaluated on five different regions R1, R2, R3, R4 and R5. Each was larger than than the next, ordered by their size, and extensions of the smaller regions. In all cases the query regions covered a minimum of a 2 x 2 cell grid, based on the 4SA's designed objective for window and containment queries.

For all regions R1-R5, the number of IOIs processed in each query execution and the query response times were obtained. The IOIs processed in each query execution were obtained to deepen the understanding of 4SA in query processing. The authors hypothesize that as 4SA reduced IOI storage in grid cells, the total number of IOIs processed in a specific query would be less compared to the absence of 4SA for the same region. The response times for window queries were obtained by considering the images where the image footprints either: (i) fully overlap, (ii) partially overlap, or were (iii) fully contained within the query regions. For containment queries, only the images that were fully contained within the respective region were considered. The results obtained for window and containment queries were similar. Thus, only the results obtained with respect to window queries are presented herein, as shown in Table 2.

4.5.1 Impact on IOIs processing

Table 2 demonstrates that the use of 4SA contributed to a 42% reduction in the IOI processing for all tested queries. In other words, compared to the total number of IOIs stored and processed by non-4SA indexes, 4SA indexes only stored and processed a 42% of IOI elements. Thus, in return 4SA contributed to a 58% improvement. The 42% reduction for all regions could be attributed to the fact that all of the bigger regions were centered around preceding smaller regions. However, further investigations is required prior to any generalization. Nevertheless, the hypothesis that the use of 4SA requires fewer IOIs to be processed is demonstrated in Table 2.

Table 2. Impact of 4SA on processing of IOIs on window queries

Region	Images records retrieved	Total IOIs processed without 4SA	Total IOIs processed with 4SA	Reduction (%)
R1	22,397	646,800	271,473	42.0
R2	43,939	1,343,066	565,408	42.1
R3	85,926	2,630,048	1,107,689	42.1
R4	177,092	5,791,178	2,436,790	42.1
R5	234,228	7,709,768	3,245,863	42.1

4.5.2 Impact on CPU execution time in query processing

Spatial query execution is computationally-intensive (Simion et al., 2012), thus, demands more computing time or CPU execution time. As 4SA reduced the IOI duplication (cf. Table 1), and processing of IOI elements (cf. Table 2), it was hypothesized that 4SA affects the CPU execution time on the employed query algorithm. To test the impact of 4SA on CPU execution time, R1 - R5 queries were executed for a single client scenario by measuring CPU time for 4SA applied and non-4SA applied

experimental settings. Each query was executed 25 times to obtain an average CPU execution time. The average CPU times for R1 - R5 queries are presented in Table 3. The results in Table 3 demonstrate that 4SA reduced the CPU execution time for the employed query algorithm by 3.0% - 5.2% (note that the NYU's Peel cluster employs high-end CPUs (cf. Section 4.2)). This reduction in CPU time is primarily attributed to the reduction in IOI storage and processing for each query.

Table 3. Impact of 4SA on CPU execution time

Region	Avg: CPU time without 4SA (miliseconds)	Avg: CPU time with 4SA (miliseconds)	Reduction (%)
R1	1373	1301	5.2
R2	2145	2082	3.0
R3	3505	3337	4.8
R4	6881	6644	3.4
R5	9615	9334	3.0

4.5.3 Impact on query response times

For each region R1-R5, window and containment query, response times were obtained. To obtain statistically robust results, each query for the R1-R5 regions was executed 25 times. As the impact of IOIs processing for each R1-R5 region was significant (i.e. 42% reduction or 58% improvement), the initial assessment on the query response times were preformed only considering the data sizes. More specifically, the experiments performed on querying of the R1-R5 regions by one client/user. The initial objective was to investigate the impact of 4SA on managing retrieval of large numbers of image records. The obtained percentage reduction on average query response times for R1 - R5 regions were respectively 0.5%, 2.6%, 0.6%, 1.9%, and 0.9%. This indicated that even for a single client, the average query response time when adopting 4SA indeed had a positive impact. Thus, adoption of 4SA for SFC-based grid cell space indexing was promising. Therefore, to demonstrate the impact of 4SA on the scalability of querying, more tests were performed. The objective of the subsequent tests was to demonstrate the ability of 4SA in managing large data sets with many concurrent users.

Data scalability in unison with user scalability evaluation was performed for the R1-R5 regions with multiple concurrent clients. The concurrent clients were simulated by multi-threading. Initially, the accessing of the R1-R5 regions by concurrent clients was performed under two scenarios. In the first, a pool of 64 threads were created and simulated concurrent users of 2, 4, 8, 16, 32, to a maximum of 64 concurrent users. In the second scenario, each time "x" number of threads and "x" number of concurrent users were created. For example, for R1, 16 concurrent threads and 16 concurrent users were created. In another instance, 32 concurrent threads and 32 concurrent users created. This was performed for all R1-R5 regions with 16, 32, 48, and 64 thread pool sizes and for corresponding 16, 32, 48, and 64 concurrent users. The maximum number of threads to be created in the thread pool were based on the number logical CPUs had in the log-in node, which was 96. Also, when calculating query times, the times incurred on thread pool creation were excluded. The effectiveness of employing 4SA on query response times were evaluated by obtaining the average query response time between the 4SA applied query times and the non-4SA applied query times. The difference in average response times was considered as the *time improvement* in Figure 7 and Figure 8.

Figure 7 and Figure 8 indicate that for larger regions (i.e.. R3, R4, and R5), and for large number of concurrent users, the time

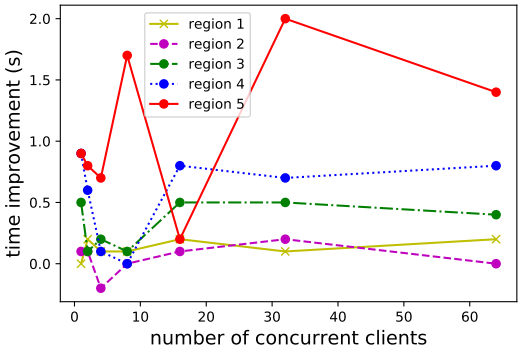

Figure 7. Scenario 1: 64 threads in thread pool

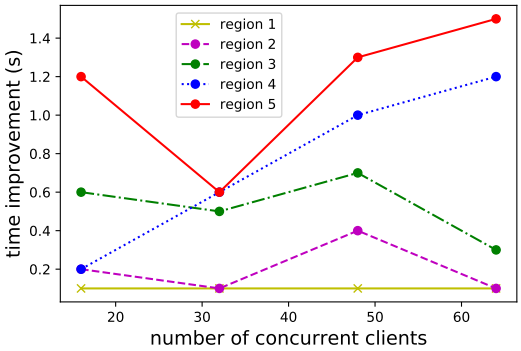

Figure 8. Scenario 2: Concurrent execution of "x" threads and "x" clients

improvements range from several hundreds of milliseconds to several seconds. In particular approximately 2, and 1.4 seconds of improvements when we created 64 concurrent threads and executed 32 concurrent users (cf. Figure 7) and 64 concurrent users (cf. Figure 8) for R5. This improvement can be attributed to the reduction in the number of IOIs to be processed for the query. This improvement indicates application of 4SA has clear benefits on searches within index structures for larger data sizes and greater numbers of concurrent users. To further demonstrate the better scalability of 4SA in terms of querying large data sizes with large number of concurrent users, two additional experiments/ tests were performed. In these two additional tests, only the three largest regions (R3, R4, and R5) were considered. Another objective of these additional tests was to understand the impact on index searching when number of concurrent users exceeds the maximum number of threads in the pool. Thus, in the first test, a pool of 32 threads created and a total of 128 concurrent users were tested. In the second test case, a pool of 64 threads created and a total of 256 concurrent users were tested. The obtained results are presented in Table 4.

The results for experiment one in Table 4 shows that the overall average time reduction for 128 concurrent users to query regions R3, R4, and R5 regions were 0.9 s (0.70%), 2.6 s (1.22%), and 5.4 s (1.86%). The reduction in query times for R3, R4, and R5 regions when employing 256 concurrent users were 2.8 s (1.07%), 7.4 s (2.40%), and 13.2 s (3.35%). This further indicates that when the number of concurrent users increased for larger regions, 4SA had a greater impact when searching within the indexes. Thus, data scalability and concurrent user scalability was further demonstrated.

Table 4. Index searching times for large query regions with large numbers of concurrent clients

Test	Region	Avg: query time without 4SA (seconds)	Avg: query time with 4SA (seconds)	Reduction (%)
1	R3	128.2	127.3	0.70
	R4	213.5	210.9	1.22
	R5	279.2	274.0	1.86
2	R3	263.0	260.2	1.07
	R4	309.3	301.9	2.40
	R5	393.8	380.6	3.35

5. CONCLUSIONS

This paper introduced an algorithmic optimization for SFC-based grid cell space indexing for RSID management within KV databases. The proposed optimization involved four steps, thus it is named the "four step algorithm" (4SA). The objective of 4SA was to store the minimum number of IOIs within KV databases for each image so the overall storage and querying performance can be improved. The 4SA mainly focused on window and containment queries where the query region overlaps with at least a 2 x 2 cell grids. Each sequential step in the 4SA focused on a unique SC type of the ground footprint of the RSID, which progressively decreased the candidate cells.

Experiments on real-world and synthetic data sets demonstrated that 4SA reduced the IOI storage by 41.3% - 47.8% compared to the non-4SA applied (i.e. original straightforward approach) SFC-based grid cell space index method. Furthermore, 4SA demonstrated data scalability for both increasingly large data sets and greater numbers of concurrent users. Specifically, due to the storage of fewer IOI in grid cells, the queries processed 42% less IOI elements. As a result, the CPU time required for data querying reduced by 3.0% - 5.2%. Additionally, this reduction had a net positive impact when querying regions with concurrent clients. The impact was significant when querying larger regions with larger numbers of concurrent clients. Specifically, when 128 and 256 concurrent users attempted to retrieve 234,228 records, 4SA reduced the index searching by 5 and 13 seconds demonstrating 1.86% and 3.35% reduction in total index search time.Thus, in overall 4SA also demonstrated user scalability and data scalability. Some notable future directions arising from 4SA work are:

- Each task in each step in 4SA can be executed in parallel. Thus, parallelization of tasks in each step will be investigated in the future.
- It is intend to explore the impact of changing the order of steps and comparing the accuracy of the different result sets.
- Apply 4SA to other non-point spatial objects that spread across multiple cells (e.g. lakes, buildings etc.) and analyze their impact on storage and query performance.

ACKNOWLEDGMENTS

This publication originated from research supported in part by a grant from Science Foundation Ireland under Grant number SFI - 17US3450. Further funding for this project was provided by the National Science Foundation as part of the project "UrbanARK: Assessment, Risk Management, Knowledge for Coastal Flood Risk Management in Urban Areas" NSF Award 1826134, jointly funded with Science Foundation Ireland (SFI - 17US3450) and Northern Ireland Trust (Grant USI 137). The clusters used for the testing were provided by NYU High Performance Computing Center. The aerial image data of Dublin were acquired with funding from the European Research Council [ERC-2012-StG-307836] and additional funding from Science Foundation Ireland [12/ERC/I2534].

References

Baumann, P., Furtado, P., Ritsch, R., Widmann, N., 1997. The rasdaman approach to multidimensional database management. *Proceedings of the 1997 ACM symposium on Applied computing*, 166–173.

Böhm, C., Klump, G., Kriegel, H.-P., 1999. Xz-ordering: A space-filling curve for objects with spatial extension. *International Symposium on Spatial Databases*, Springer, 75–90.

Gaede, V., Günther, O., 1998. Multidimensional access methods. *ACM Computing Surveys (CSUR)*, 30(2), 170–231.

Huang, S., Wang, B., Zhu, J., Wang, G., Yu, G., 2014. R-hbase: A multi-dimensional indexing framework for cloud computing environment. *2014 IEEE International Conference on Data Mining Workshop*, IEEE, 569–574.

Hughes, J. N., Annex, A., Eichelberger, C. N., Fox, A., Hulbert, A., Ronquest, M., 2015. Geomesa: a distributed architecture for spatio-temporal fusion. *Geospatial Informatics, Fusion, and Motion Video Analytics V*, 9473, International Society for Optics and Photonics, 94730F.

Jing, W., Tian, D., 2018. An improved distributed storage and query for remote sensing data. *Procedia Computer Science*, 129, 238–247.

Kafando, R., Decoupes, R., Sautot, L., Teisseire, M., 2020. Spatial Data Lake for Smart Cities: From Design to Implementation. *AGILE: GIScience Series*, 1, 1–15.

Laefer, D., Abuwarda, S., Vo, A., Truong-Hong, L., Gharibi, H., 2017. 2015 Aerial Laser and Photogrammetry Survey of Dublin City Collection Record. (Last accessed by 20/10/2019).

Papadopoulos, S., Datta, K., Madden, S., Mattson, T., 2016. The TileDB array data storage manager. *Proceedings of the VLDB Endowment*, 10(4), 349–360.

Samet, H., 2006. *Foundations of multidimensional and metric data structures*. 1^{st} edn, Morgan Kaufmann, San Francisco, CA, USA.

Simion, B., Ray, S., Brown, A. D., 2012. Speeding up spatial database query execution using GPUs. *Procedia Computer Science*, 9, 1870–1879.

Wang, L., Cheng, C., Wu, S., Wu, F., Teng, W., 2015. Massive remote sensing image data management based on hbase and geosot. *2015 IEEE international geoscience and remote sensing symposium (IGARSS)*, IEEE, 4558–4561.

Wang, L., Yan, J., Ma, Y., 2019. *Cloud computing in remote sensing*. CRC Press.

Zhou, X., Wang, X., Zhou, Y., Lin, Q., Zhao, J., Meng, X., 2021. RSIMS: Large-Scale Heterogeneous Remote Sensing Images Management System. *Remote Sensing*, 13(9), 1815.