DACMA: Designing space ordering optimizations to scalably manage aerial images

1st Chamin Nalinda Lokugam Hewage
School of Computer Science
University College Dublin
Dublin, Ireland
chamin.lokugamhewage@ucdconnect.ie

2nd Debra Laefer
Center for Urban Science and Progress
New York University
New York, USA
debra.laefer@nyu.edu

3rd Michela Bertolotto
School of Computer Science
University College Dublin
Dublin, Ireland
michela.bertolotto@ucd.ie

4th Anh Vu Vo
School of Computer Science
University College Dublin
Dublin, Ireland
anhvu.vo@ucd.ie

5th Nhien-An Le-Khac
School of Computer Science
University College Dublin
Dublin, Ireland
an.lekhac@ucd.ie

2022 IEEE International Conference on Big Data (Big Data) | 978-1-6654-8045-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/BigData55660.2022.10020833

***Abstract*—Aerial images are a special class of remote sensing images, as they are intentionally collected with a high degree of overlap. This high degree of overlap complicates existing index strategies such as R-tree and Space Filling Curve (SFC) based index techniques due to complications in space splitting, granularity of the grid cells and excessive duplication of image object identifiers (IOIs). However, SFC based space ordering can be modified to provide scalable management of overlapping aerial images. This involves overcoming similar IOIs in adjacent grid cells, which would naturally occur in SFC based grids with such data. IOI duplication can be minimized by merging adjacent grid cells through the proposed "Designing Adjacent Cell Merge Algorithm" (DACMA). This work focuses on establishing a proper adjacent cell merge metric and merge percentage value. Using a highly scalable, distributed HBase cluster for both a single aerial mapping project, and multiple aerial mapping projects, experiments evaluated Jaccard Similarity (JS) and Percentage of Overlap (PO) merge metrics. JS had significant advantages: (i) generating smaller merged regions and (ii) obtaining over 21% and 36% improvement in reducing query response times compared to PO. As a result, JS is proposed for the merge metric for DACMA. For the merge percentage two considerations were dominant: (i) substantial storage reductions with respect to both straight forward SFC-based cell space indexing and 4SA based indexing, and (ii) minimal impact on the query response time. The proposed merge percentage value was selected to optimize the storage (i.e. space) needs and response time (i.e. time) herein named the "Space-Time Trade-off Optimization Percentage" value (or STOP value) is presented.**

***Index Terms*—aerial images, space filling curves, scalability, space-time trade-off**

I. Introduction

The rise in the affordable imaging cameras and image acquisition modes is resulting in a profound growth in Remote sensing images (RSIs) [1]. Consequently, the development of RSIs management systems that can cope with increased RSIs volumes, as well as traffic volumes (i.e. scalable), while also achieving commensurate query performance is a research topic of increasing interest. Today, most research in highly-scalable, RSIs management systems is happening through the innovation of novel spatial indexes and index optimizations. Typically, these innovations are targeted at improving window and containment queries - queries that retrieve images or image tiles, which either fully or partially overlap and/or completely reside within a specific region. Ideally, these are coupled with databases that are designed specifically to manage increased data and traffic volumes via the distribution of loads across multiple, horizontally-scaled machines.

Arguably, the most seminal research has included "raster data manager" (RASDAMAN) [2], "remote sensing image management system" (RSIMS) [3], GeoMesa [4], and TileDB [5], [6] and [7]. To accommodate increased volumes of RSIs and potential growth in users, RSIMS, RASDAMAN and TileDB employ horizontally-scaled, object relational databases. GeoMesa and systems from [6] and [7] are built atop Key-Value (KV) databases - a class of NoSQL database that is inherently scalable for high data demands and user demands. In their indexing, both RASDAMAN and TileDB implement R+-tree and R-tree indexes by organizing the minimum bounding rectangles (MBRs) of RSIs. RSIMS and the aforementioned KV database oriented systems implement Space Filling Curve (SFC) based indexes.

Traditionally, application of R-tree and its variants on imagery data have been performed by organizing two-dimensional (2D) MBRs of images into overlapping spatial regions. Successful deployment of the R-tree and its variants requires recognizition of overlapping spatial subdivisions. In contrast, SFC-based indexing employs space ordering techniques that divide the space into distinct levels of grids and organize grid cells of uniform spatial resolution in each grid level according to the respective SFC value (e.g. a Hilbert value or Morton code). This approach organizes images or image tiles into a specific grid cell at a particular grid level according to the spatial coverage of the image. For aerial images, the adoption of both the R-tree or its variants and SFC-based indexes can pose significant challenges.

Aerial images are a special class of RSIs that involve flight paths, as well as camera configurations designed to ensure a great deal of overlapping (e.g. 20-80%) among adjacent images (see Section IV for more details). Figure 1 demonstrates the high degree of overlap among images captured in an area of $2km^2$ during the 2015 LiDAR and imagery mapping exercise performed in the city of Dublin, Ireland [8]. In Figure 1, each green color squares represent the bounding box of an aerial image. Figure 2 also illustrates the substantial number of images that may overlap with a specific geographic region. As depicted in Figure 2, the superimposing of a 64 x 64 grid (where each unit cell has a height and width of 60 m) quantifies the images overlapping with each grid cell.

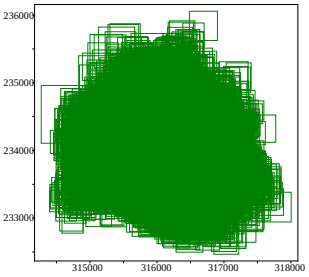


Fig. 1: High degree of overlap among images

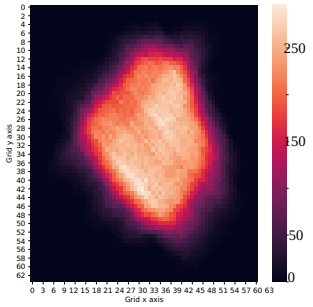


Fig. 2: Heat-map of image overlapping

These inherent characteristics of aerial images, especially the high degree of overlap among images, poses challenges in using R-tree type indexes. Generally, adoption of R-tree requires identification, of overlapping spatial subdivisions of the region. The massively overlapping of aerial imagery complicates the splitting of the space into distinct overlapping regions. Furthermore, identifying the potentially optimum subdivision is cumbersome. Finally, a traditional R-tree like index structures limits the possibility of adopting highly scalable databases such as KV databases as an R-tree would require maintenance of an in-memory index. This is an impediment from scalability perspective in managing aerial images at scale.

Similarly, current SFC-based indexes also exhibit substantial challenges when being used to index aerial images. For example, when adopting a SFC-based grid, the high degree of image overlap causes a significant amount of overlap with the cell boundaries. Traditionally this has resulted in a much coarse grid. However, larger grid cells may not represent the actual image coverage and would require filtering more images during the querying process.

To mitigate those challenges, an SFC-based indexing can be tailored for aerial images by using a single level SFC grid. Such a solution would store the image object identification (IOI) of each image in all the cells that the image overlaps [9]. As several images may overlap with a given cell, such a solution would store multiple IOIs in each distinct cell. Once the IOIs for each cell are determined, the spatially ordered cells can be stored in a highly-scalable, clustered database environment, such as a KV database or a relational database. When stored database, the associated SFC value (e.g. Hilbert value or Morton code) for each cell will be the key in the respective database table. The corresponding IOI list for each cell will be stored as the value for each key in the database table record.

While such a solution for aerial image management is straight forward and scalable, it has limitations. The main one is the necessity to store the IOI of each aerial image in every grid cell for all images for which there is an overlap. Arguably, unless the size of the unit grid cell is coarse, the storage of the IOIs across every cell covering each RSIs would cause excessive storage consumption [10]. Consequently, it would also require excessive IOI writes as well as updates, if subsequent updates are necessary for IOIs. In the above solution, the IOIs of adjacent cells share a high degree of similar IOIs. This opens the avenue for "Designing Adjacent Cell Merge Algorithms (DACMA)" for SFC-based index optimizations for scalably-managed, aerial images.

As such, this work studies two critical parameters relevant to DACMA. The first is the merge metrics for combining adjacent cell; this work considers "Jaccard Similarity (JS)" and "percentage of overlap (PO)" merge metrics. The second is the merge percentage, which examines the percentage of similarity of the IOIs between the adjacent cells. The selection of a merge percentage has the potential for substantial impact on overall storage performance and window/ containment query performance. Thus, investigation of merge percentages in tandem with storage reductions and query times are considered for DACMA. Through the utilization of two synthetically generated data sets, this work identifies the merge metric and the merge percentage. The pivotal contributions of DACMA can be summarized as follows:

- Introduction of DACMA as a novel algorithm for scalable aerial image data management.
- Evaluation of adopting JS and PO for DACMA in space splitting, by forming merged grid cells (i.e. blocks) and query response times for similar storage reductions.
- Confirmation of JS as a better merge metric for DACMA.
- Identification of a merge percentage termed **STOP value** which yields substantial storage reduction with only minimal impact on query time, with the aid of the previously introduced "Four Step Algorithm (4SA)" [11].

II. Related Work and 4SA

A. State-of-the-art RSIs management systems

RASDAMAN [2] is a leading RSIs management system. RSIs in RASDAMAN are stored as binary large objects inside a PostgreSQL database. To index images, RASDAMAN employs R+-tree and GiST indexes. TileDB [5] is another RSIs management system that employs R-tree. As stated, application of R-tree or its variants require the spatial regions to divide into overlapping regions. However, due to its inherent nature, its cumbersome to identify distinct overlapping regions for aerial images. Thus, use of RASDAMAN or TileDB for aerial image management will require extensive investigation

of identifying appropriate spatial splittings as well as woul also require to consider larger spatial regions. This is a impediment to scalable management of aerial images.

The RSIMS [3] is a RSIs data management system built ato a PostgreSQL database. In indexing images, RSIMS employ a distributed multi-level, Hilbert index. When indexing image or image tiles, RSIMS determines the grid level at which th spatial indexes needs to be calculated. This level is based o the spatial coverage of the respective image. Once the leve is identified, the corresponding Hilbert index for the image i calculated through the latitude and longitude of the geometri centre of the image. Admittedly, this approach requires the im age to be completely contained within a grid cell. Nevertheless as aerial images have high degree of overlap, superposition o a grid could result in a substantial amount of images not full contained within grid cell. Thus RSIMS's index strategy is nc feasible in indexing aerial images.

Similar challenges arise in the adoption of [6] and [7] which are built atop KV databases. The is because their inde strategies also employ multi-level grid systems and SFC-base indexing to spatially organize imagery data – thus requiring th RSIs to precisely fit into grid cells. This complicates indexin where there are a substantial number of aerial images.

GeoMesa is another leading RSIs management system buil atop a KV database. The focus of GeoMesa is to inde image tiles, not raw images. To index tiles, GeoMesa employs XZ-SFC [10]. When a given spatial object overlaps with multiple cells, the XZ-curve forms a much larger region by increasing the height and width of the bottom left cell by a factor of 2 (upwards and to the right). Although this approach theoretically works for any object with a spatial extent, the authors did not discuss application of this concept for non point objects that have a high degree of overlap. For much larger regions, employing an *extended* region concept would ultimately result in a single, massive grid cell. Therefore adoption of XZ-SFC curve will work for indexing image tiles it would prohibit the indexing the well formatted real world images that have high degree of overlap as well as raw non axis align aerial images that also have a significant overlap.

B. Four Step Algorithm (4SA)

The Four Step Algorithm or 4SA [11] is for optimizatior of an SFC-based grid cell space indexing to scalably manage RSIs in KV databases. The 4SA was specifically designed fo window and containment queries where the region of interes overlaps with at least a 2 x 2 grid of cells. The objective o 4SA was to avoid the IOI storage in grid cells if a particula grid cell exhibits a certain spatial configuration (SC) type with respect to its adjacent grid cells in its overall image coverage The four SC types of 4SA are presented in Figure 3.

In step 1, 4SA checks if the SC type "+" exists, for the IO of the image for its currently considered cell. If the SC type "+" exists, then 4SA omits storage of its IOI in the respective cell. Similarly, in steps 2-4, the 4SA check if the other SC types: "X", "Ts", and "ls" for currently considered cell of the respective image. If existing, the respective IOI of the image

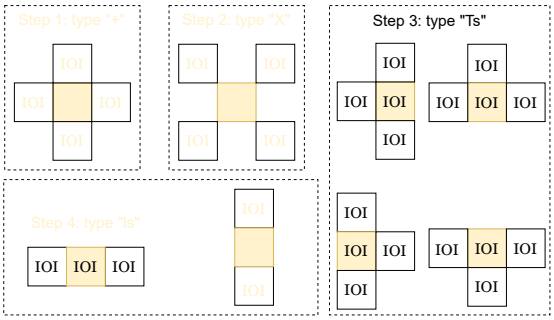


Fig. 3: Spatial configuration types in each step in 4SA

will not be stored in the currently considered cell. As a result, application of 4SA to a set of RSIs reduces the overall storage cost (as well as the write/ update costs). Additionally, due to the reduced number of IOIs, the processing, window, and containment query times are also reduced.

III. Methodology

A. Intuition of DACMA

Figure 4 demonstrates the intuitiveness of DACMA. In Figure 4, the letters 'A'-'L' represent IOIs of 12 images that overlap with the grid space. In the case of a non-DACMA scenario (i.e. directly adopting straight SFC based space indexing), 12 IOIs are stored 24 times in total across the 8 grid cells. In contrast, when the adjacent cells are merged based on the condition if there's an overlap of 80% or more among adjacent cells, the total number of cells are reduced

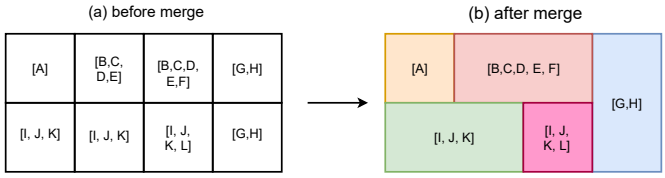


Fig. 4: Merge adjacent cells if IOIs overlap by 80%

Another notable feature of the merge process is the total number of images in the blocks. For example, the second block in the first row contains 5 IOIs. Prior to the adjacent cell merge, the second and the third cells in the first row contained only 4 and 5 IOIs respectively. Therefore, if a user executes a window/ containment query with in the region of cell 2 in the first row, it would only require evaluation of 4 images. However, after the cells merge, the same query requires evaluation of 5 images. Therefore, although adjacent cell merging reduced the storage, in some instances, it can impact the query times negatively. However, reduction of IOI storage also indicates a reduction of the IOI writes/ updates required to access the database tables. Thus, DACMA produces both storage reductions and database write/ update reductions while potentially impacting query time.

B. Adjacent cell merge metrics (i.e. $mergeMetric$)

In the above example, the adjacent cells were merged based on "Jaccard similarity" (JS) among the IOIs in the adjacent cells. Equation 1 details the JS metric. Suppose the C1 cell and the C2 cell are adjacent and that each cell has a specific number of IOIs stored among the cells and demonstrate a degree of overlap. In such scenarios, the JS *mergeMetric* defined as the percentage of ratio between the size of the IOIs in the intersection of C1 and C2 with the total number of distinct elements in the union of C1 and C2. If the ratio exceeds a defined threshold (i.e. *mergePercentage*), the adjacent cells are merged.

In addition to the JS, assessing *similarity* between the IOIs in adjacent cells can be done through alternative approaches. One such alternative would be to first check the size of the intersection between the common IOIs in the adjacent cells. Subsequently, merge would only occur, if the ratio between the size of the intersected elements in the two cells and size of the currently considered cell (or already merged block) exceeds a certain threshold. This *mergeMetric* in this research work is defined as "percentage of overlap" (PO). Equation 2 shares the formula for the PO metric.

$$JS = \left\{ \frac{|C1 \cap C2|}{|C1 \cup C2|} * 100\% \right. \tag{1}$$

$$PO = \left\{ \frac{|C1 \cap C2|}{|C1|} * 100\% \right. \tag{2}$$

C. Explanation of DACMA

Algorithm 1 outlines DACMA. Algorithm 1 employs two input parameters. The first is a SFC-based spatially organized grid (i.e. $\{SOGrid\}$). Each cell in the $\{SOGrid\}$ is comprised of the list of IOIs of the images for which each cell is overlapping. The second input parameter is the merge percentage (i.e. *mergePercentage*). The *mergeParameter* is the minimum percentage that must consider in performing an adjacent cell merge. After successfully processing the $\{SOGrid\}$ subjected to the minimum *mergeMetric*, DACMA produces two outputs: (i) a set of blocks with the corresponding IOIs included in each block (i.e. $\{MrgBlksWithImgs\}$) and (ii) the set of distinctly aggregated sets of grid cells (i.e. $\{MrgSOValsBlks\}$).

At the start, DACMA, defines five empty sets (Line 2-6), which includes $\{MrgBlksWithImgs\}$ and $\{MrgSOValsBlks\}$ sets to produce the output of DACMA. The other three sets are defined to facilitate three requisite intermediate steps: (i) a set that stores the images in the current merge process (i.e. $\{currMrgPr\}$; (ii) a set that stores the spatially ordered values involved in the current merge process (i.e. $\{currMrgPrSOVals\}$), and (iii) a set that manages the grid cells with no images (i.e. $\{noImgRegions\}$).

Once DACMA defines the initial sets, it loops through each grid cell set (i.e. $\{SOGridCell\}$) in $\{SOGrid\}$. For each $\{SOGridCell\}$, DACMA obtains the corresponding spatial order value (i.e. $SOVal$ such as a Hilbert value) and the

Algorithm 1: DACMA

Input: $\{SOGrid\}$ ▷ spatially ordered grid set
$mergePercentage$
Output: $\{MrgBlksWithImgs\}$ ▷ set of merged blocks with images in the blocks
$\{MrgSOValsBlks\}$ ▷ set of merged blocks with cell ids that each block contains

1 **Function** DACMA($\{SOGrid\}, mergePercentage$)**:**
2 $\{currMrgPr\} \leftarrow \emptyset$
3 $\{currMrgPrSOVals\} \leftarrow \emptyset$
4 $\{noImgRegions\} \leftarrow \emptyset$
5 $\{MergBlksWithImgs\} \leftarrow \emptyset$
6 $\{MergSOValsBlks\} \leftarrow \emptyset$
7 **foreach** $\{SOGridCell\} \in \{SOGrid\}$ **do**
8 $SOVal \leftarrow \{SOGridCell\}$
9 $\{imgLst\} \leftarrow \{SOGridCell\}$
10 **if** $\{imgLst\} \neg \emptyset$ **then**
11 **if** $|\{currMrgPr\}| == 0$ **then**
12 $\{currMrgPr\} = \{imgLst\}$
13 $\{currMrgPrSOVals\}_{add}(SOVal)$
14 **end**
15 **else**
16 $\{\chi\} = \{currMrgPr\} \cap \{imgLst\}$
17 $\{\nu\} = \{currMrgPr\} \cup \{imgLst\}$
18 $JS = (|\{\chi\}|/|\{\nu\}|) * 100$
19 $PO = (|\{\chi\}|/|\{imgLst\}|) * 100$
▷ $mergeMetric$ - *i.e. either JS or PO*
if
$mergeMetric >= mergePercentage$
then
$\{currMrgPr\} += \cup\{imgLst\}$
$\{currMrgPrSOVals\}_{add}(SOVal)$
end
else
CompleteCurrentMerge()
end
20 **end**
21 **end**
22 **else**
23 $\{noImgRegions\}_{add}(SoVal)$
24 **if** $|\{currMrgPr\}| != \emptyset$ **then**
CompleteCurrentMerge()
25 **end**
26 **end**
27 **end**
28 ***return***
$\{MrgBlksWithImgs\}, \{MrgSOValsBlks\}$
End Function
30 **Function** CompleteCurrentMerge()**:**
31 **return**
32 $\{MrgBlksWithImgs\}_{add}(\{currMrgPr\})$
33 $\{MrgSOValsBlks\}_{add}(\{currMrgPrSOVals\})$
34 $\{currMrgPr\} \leftarrow \emptyset$
35 $\{currMrgPrSOVals\} \leftarrow \emptyset$
36 $\{currMrgPr\} = \{imgLst\}$
37 $\{currMrgPrSOVals\}_{add}(SOVal)$
38

set of the IOIs of the images that overlap with the corresponding $\{SOGridCell\}$ (i.e. $\{imgList\}$. Subsequently, for each $\{imgList\}$, DACMA checks if it is empty (meaning no images overlap with the cell) or not empty. If $\{imgList\}$ is not empty, DACMA searches for an on-going merge process by checking $\{currMrgPr\}$. At the process's onset, there are no on-going merge processes. Hence, the size of $\{currMrgPr\}$ is zero (Line 11). Thus, when the $\{imgList\}$ is not empty and DACMA is in its early execution steps, the $\{currMrgPr\}$ is initialized with $\{imgList\}$. Importantly, the $SOVal$ of the corresponding cell is added to the $\{currMrgPrSOVals\}$ (Line 12-13).

In circumstances where there is an on-going merge, DACMA builds the corresponding merge metric- [i.e. either JS or PO (Line 16-19)]. This obtains the intersection between the current image set and all the images in the on-going merge process (i.e. $\{\chi\}$) in-unison the union of current image set and the ongoing image set (i.e. $\{\nu\}$). Development of PO is performed by considering the intersection between the current image set and all the images in the on-going merging and the size of the current image set (i.e. $|\{imgLst\}|$). If the value obtained for the intended *mergeMetric* satisfies the minimum *mergePercentage*, then the current $\{imgList\}$ is aggregated with the images in the current merge process. The corresponding $SOVal$ is also added to the $\{currMrgPrSOVals\}$.

When the *mergeMetric* is less than the *mergePercentage*, the current merge process is completed through the execution of the $CompleteCurrentMerge$ method (Line 31-37). This adds the already existing $\{currMrgPr\}$ and $\{currMrgPrSOVals\}$ in the on-going merge process to the two sets that DACMA returns: $\{MrgBlksWithImgs\}$ and $\{MrgSOValsBlks\}$. Once the on-going merge process is completed, a new merge process must be reinitiated for the subsequent adjacent cell merging. Thus, both $\{currMrgPr\}$ and $\{currMrgPrSOVals\}$ are set to null ($\emptyset$) and initialize the $\{imgList\}$ and $SOVal$ with respect to the $\{SOGrid\}$.

In fact, $CompleteCurrentMerge$ is also required to be performed when there are no images in the adjacent cells, while there is a non-empty $\{currMrgPr\}$ (Line 24). Furthermore, when there are no images in the $\{SOGrid\}$, DACMA adds the corresponding $SOVal$ of the cell to the "no image regions" set (i.e. $\{noImgRegions\}$). Once the entire $\{SOGrid\}$ is processed, DACMA returns the two output result sets.

IV. EXPERIMENTAL EVALUATION

A. Study data

All experiments in DACMA employed a 2015 real world, aerial imagery data set covering 2km^2 of Dublin Ireland's city centre in [8]. The data set inlcudes 8,438 images. Each has a bounding box of approximately 400 m x 400 m (cf. Figure 1). The imagery primarily consists of two classes: (i) oblique images and (ii) true colored (i.e. RGB) and infrared (i.e. CIR) images. Both the RGB and CIR images were acquired from a nadir camera that was mounted perpendicular to the aerial mapping platform - thus the camera was directly facing the earth. The oblique images were taken from two tilted cameras that were set at 30° degrees tilt from the mapping platform.

Notably, this mapping exercise included 41 overlapping flight paths spaced at 100 meters intervals. Each flight operated at a 300 meter altitude, and each flight path was oriented 45° degrees to Dublin's dominant street grid. The 45° degree orientation was deployed to alleviate the self-shadowing effect and to maximize data coverage of building facades. The camera configurations and flight path configurations were established intentionally to introduce a significant degree of overlap between successive aerial images along a flight line (i.e. *forward lap*), as well as overlap between successive photos from adjacent fight lines (i.e. *side lap*). With such a configuration, the surveyed 2km^2 was scanned 6 times from multiple angles, while producing many images overlap of the given geographic areas.

The configuration of the aforementioned real world dataset was used herein to generate two larger synthetic data sets to evaluate DACMA's performance. The first synthetic data set was designed to be indicative of a single airborne mapping project corresponding to surveying one region of 100 km^2 (i.e. 10 km x 10 km). The resulting heat-map of aerial image distribution was obtained by super imposing a grid where each unit cell had a width and height of 60m and is presented in Figure 5. The second synthetic data set was designed correspond to multiple airborne mapping projects around the same 10 km x 10 km region. As a result, the super imposition of same grid atop second simulated data set resulted in multiple clusters as shown in an image distribution heat-map Figure 6. In each synthetic data sets a total of 250,000 image data records for each class were generated.

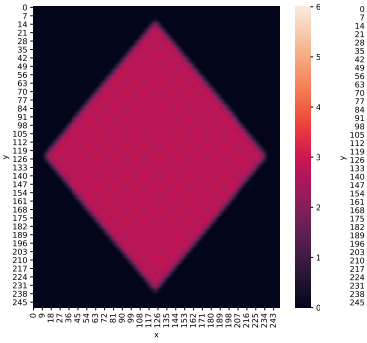


Fig. 5: Heat-map of the single cluster

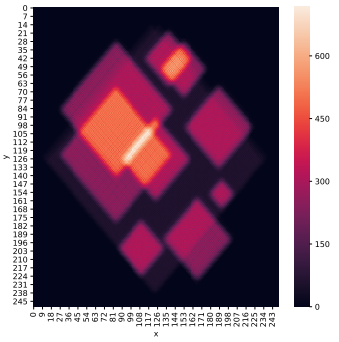


Fig. 6: Heat-map of multiple clusters

B. Experimental setup

The experiments were design to identify a proper merging metric and evaluation of scalability. The generation of the synthetic data sets, and the initial application of DACMA were performed locally. Subsequent DACMA experiments were performed on an inherently scalable HBase KV database in the form of the Peel cluster at New York University (NYU). Peel is a high-end, 18 node cluste,r which has deployed HBase (HBase 2.1.0) atop Hadoop (3.0.0).

C. Selecting proper merge metric

The rationale of DACMA is based on the merging of adjacent cells, if the adjacent cells share a significant number of IOI. While superficially straightforward, from a deeper analytical prospect, depending on the merge metric, such a rationale can yield substantially different results. Thus, while employing JS and PO as merge metrics, multiple experiments were performed to discern a better metric that would suitable for both a single cluster and multiple cluster scenarios.

In designing the tests, first, DACMA was applied to both the synthetically generated single cluster data set and the multi-cluster data set for a range of merge percentages. For instance, for a single cluster, PO based DACMA was applied at 72.5%, 75%, 77.5%, 78.5%, 80%, and 82.5% merge percentages and the JS based DACMA was applied at 5%, 7.5%, 15%, 20%, 25%, 30%, 35%, 40%, 45%, and 50% merge percentages. Similarly, a range of merge percentages were applied for multiple clusters, for both the PO based DACMA and the JS based DACMA. As specified in Algorithm 1, the application of DACMA, resulted in two outputs: (i) a set of merged grid cells (i.e. blocks), and (ii) grid cells to block mapping set. The obtained outputs for each JS and PO based DACMA solutions for each synthetic data set were subsequently ingested into HBase tables. Thereafter, the storage consumption for each PO and JS based DACMA solutions for both single and multiple clusters scenarios were obtained. Additionally, the storage requirement for aerial images in the non-DACMA situation was also obtained.

A better merge metric was investigated by analyzing the JS and PO merge percentages that produced similar storage reductions for each synthetically generated data sets. three major factors were analyzed: (i) layout of space splittings; (ii) minimum, maximum, and average number of images in the merged grid cells (i.e. blocks) including total number of blocks; and (iii) average query response times for five arbitrarily chosen regions.

Table I contains the results from the layout of space splittings. The results obtained for minimum, maximum, and average number of images in the block with total number of blocks are presented in Table II. Analysis of average query times are outlined in Table III.

1) Layout of space splitting:

The layout of space splittings in Table I demonstrates that in scenarios where similar storage reductions occurred, JS based DACMA tended to generate smaller merged regions irrespective of whether DACMA was applied to a single cluster or multiple clusters scenario. For example, when the storage reduction was 91.4% in the single cluster scenario, smaller merged regions were clearly visible in the JS applied DACMA (i.e. JS = 7.5%) scenario as opposed to the counter part PO applied DACMA (i.e. PO = 75.0%). While similar outcomes can be seen across all instances of JS being applied to DACMA scenarios in the single cluster, this was even more evident in multiple cluster scenarios where the overall storage reduction was approximately 93.4% where the corresponding JS percentage was 1.7% and the PO percentage was 70.0%.

2) Statistics on blocks (i.e. merged cells):

Further insights on the impacts of adopting JS and PO as merge metrics can be seen in both Table II and Table I. Specifically, irrespective of the merge percentage, the number of total blocks in the JS based DACMA experiments was always less than in the PO based DACMA experiments for the same storage reductions. The reduction in blocks ranged from 20.1% to 47.0% for the single cluster scenario and 51.0% - 57.6% for the multiple cluster scenario. Admittedly, the block reduction was negatively influenced by the JS based DACMA solutions which included an increase in the average number of images in the blocks. This increase ranged from 58.1% to 84.1% and from 108.5% to 152.1% for the multi-cluster scenario.

The analysis on the maximum number of images in the blocks demonstrated similar patterns as that for the total number of blocks. Specifically, the maximum number of images that the blocks included was considerably lower for JS based DACMA solutions. This reduction ranged from 66.6% to 77.2% for the single cluster scenario and from 60.8% to 72.7% for the multi-cluster scenario. As discussed with respect to Table I, JS based DACMA solutions produce comparatively smaller merged regions compared to PO based DACMA solutions. Thus, JS based DACMA solutions are likely to host comparatively fewer images compared to PO based DACMA solutions.

Based on block reduction in Table II and comparatively smaller merged regions in Table I for the JS with DACMA solution, further insights discerned. One is the influence of JS and PO on the layout of space splittings. Specifically there were few excessively large or extremely small space splits occurring when adopting JS for DACMA compared to PO for DACMA. Visual space splittings of PO based DACMA clearly shows a large number of regions, but Table II also demonstrates a large number of total blocks. This suggests that the adoption of PO for DACMA results in comparatively large, merged regions, as well as comparatively smaller merged regions compared to JS based DACMA merged regions which are all tend to be medium in their merged sizes.

3) Comparison of average query times:

Table III presents the average query times when adopting JS and PO for DACMA for one use-case for a single cluster and multiple cluster. JS as opposed to PO for DACMA always yielded lower average query times. For the single cluster scenario, the reduction in average time ranged from 6.0% to 21.4% for the tested five regions. For the multiple cluster scenario, the reductions ranged from 2.4% to 36.6%. Although due to space constraints, only one use-case from from both single and multiple clusters scenarios are reported herein, this reduction was observed across all performed querying experiments. Therefore, the authors concluded that JS based DACMA solutions always yield better reduced query times compared to PO based DACMA solutions for similar storage reductions. Previously observed results, such as JS based DACMA solutions having smaller merged regions and the a considerably fewer maximum number images hosted in the

TABLE I: Comparison of layouts

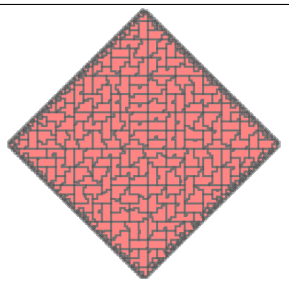
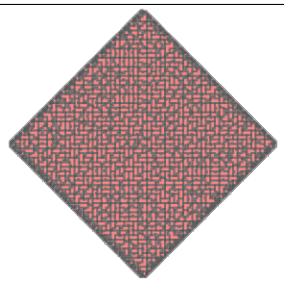
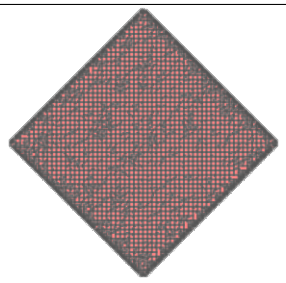
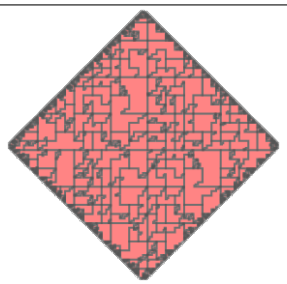
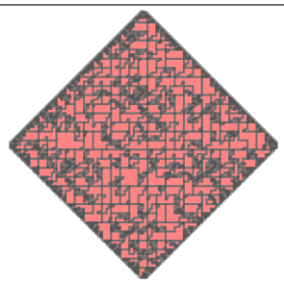
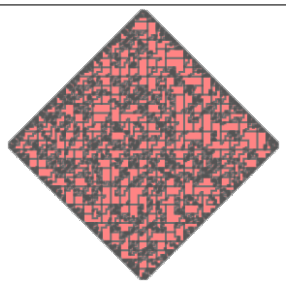
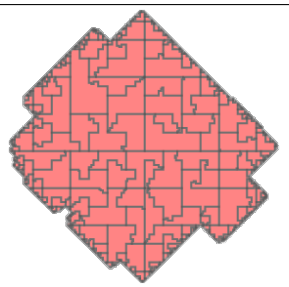
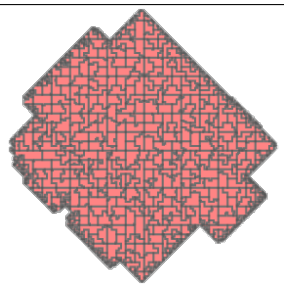
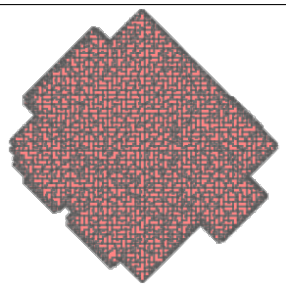
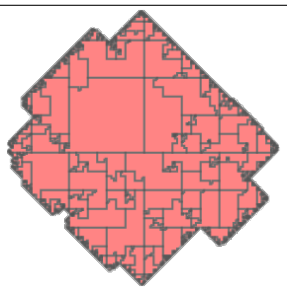
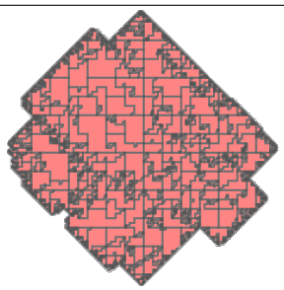
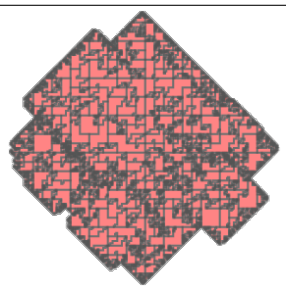
Single cluster	Storage reduction (%)		∼91.4	∼87.0	83.4
	Layout	Jaccard similarity (JS)	 JS = 7.5%	 JS = 20.0%	 JS = 30.0%
		Percentage of overlap (PO)	 PO = 75.0%	 PO = 77.5%	 PO = 78.5%
Multiple clusters	Storage reduction (%)		∼93.4	∼90.5	∼86.7
	Layout	Jaccard similarity (JS)	 JS = 1.7%	 JS = 10.0%	 JS = 20.0%
		Percentage of overlap (PO)	 PO = 70.0%	 PO = 75.0%	 PO = 77.0%

TABLE II: Analysis of aspects of Jaccard and PO for similar storage reductions

Scenraio	Storage reduction (%)	Jaccard Similarity (JS)					Percentage of Overlap					Reduction with JO (%)		Increase in avg images (%)
		%	total blocks	Images in blocks			%	total blocks	Images in blocks			total blocks	Max images in blocks	
				max	min	avg			max	min	avg			
single cluster	∼91.4	7.5	597	3,247	1	974	75	1,127	9,723	1	535	47.0	66.6	82.1
	∼87.0	20	1,356	1,267	1	691	77.5	2,148	5,569	1	437	36.9	77.2	58.1
	83.4	30	2,571	854	1	718	78.5	3,217	3,656	1	390	20.1	76.4	84.1
multiple clusters	∼93.4	1.7	386	22,433	1	1074	70	910	74,485	1	426	57.6	69.9	152.1
	∼90.5	10	819	5,781	1	793	75	1,762	14,740	1	354	53.5	60.8	124.0
	∼86.7	20	1,598	3,022	1	590	77	3,257	11,056	1	283	51.0	72.7	108.5

TABLE III: Impact of JS and PO on average query response time for similar storage reductions

	Storage reduction %	PO %	JS %	Num: images	Avg: query time (s)		Reduction in query time (%)
					PO	JS	
Single cluster	91.4	75.0	7.5	6,205	2.7	2.3	14.8
				11,993	3.9	3.4	12.8
				24,053	7.0	5.5	21.4
				48,603	9.7	9.4	3.1
				96,139	18.4	17.3	6.0
	Storage reduction %	PO %	JS %	Num: images	Avg: query time (s)		Reduction in query time (%)
					PO	JS	
Multiple clusters	93.5	70.0	1.7	10,905	13.2	8.4	36.4
				21,556	13.3	8.5	36.1
				23,070	14.1	11.5	18.4
				69,230	19.3	14.0	27.5
				121,206	25.0	24.4	2.4

blocks influenced such reductions in query times compared to the PO based DACMA.

4) Concluding remarks on merge metric:

The overall impact of JS compared to PO as a merge metrics for DACMA for similar storage reductions was independent of it being a single or multiple cluster scenarios. The JS based DACMA solutions always: (i) yielded smaller merged regions, (ii) comprised a smaller number of total blocks, (iii) hosted a larger average number of images in the blocks, (iv) hosted a substantially lower maximum number of images in the blocks, and (iv) most importantly yielded reduced average query times. Consequently, the adoption of JS as a merge metric is recommended for designing adjacent cell merge algorithms-i.e. for DACMA. Thus, the subsequent experiments investigated the avenues in designing adjacent cell merge algorithms by employing JS as the merge metric for DACMA.

D. Identify the merge percentage for JS

The experiments performed to establish JS as the better merge metric over its counterpart PO also indicated that the chosen merge percentage had a substantial impact on query performance time. As shown in Table III, when the JS percentage value increased, the average query response times decreased. The primary reason was the smaller sized merge regions that were formed due to the increased JS percentage (cf. Figure I). Consequently, the number of images that each smaller block host is small. However, as the query times decreased, the overall obtainable storage reduction compared to non-DACMA applied solutions also reduced.

Typically, when designing new spatial indexes or spatial index optimizations, one or two categories of costs associated with index design are reduced [12]. These include: (i) storage costs, (ii) response times, and (iii) write/update costs. Optimization of all three in a specific index or index optimization solution ia practically infeasible. Thus a prioritization on the optimization of cost category (or categories) must be determined in advance. As stated previously, DACMA was primarily envisioned to reduce IOI duplication among adjacent grid cells through adjacent grid cell merging. Intuitively, the reduction of IOI duplication also results in a reduction of the number of IOIs required to write into database table(s). In addition, it also reduces the number of merged cells that need to be updated in scenarios where a specific IOI is changed. Thus, admittedly, DACMA primarily reduces the storage cost in adopting SFC curved based grid cell space indexing. More over, DACMA also reduces write/update costs as a result of reduced storage of IOI. However, as anticipated, both storage cost and update/write cost reductions were achieved at the expense of query response time.

Thus, identification of an optimal JS percentage for adjacent cell merge that reduces storage cost (and write/update cost), while having a minimally adverse impact on query time is critical. This subsection describes an approach for identifying an optimal or near optimal JS percentage value to balance the impacts of DACMA on storage (i.e. *space*) cost and query response time (i.e. *time*). The optimal JS value herein called the "***Space-Time Trade-Off Optimization Percentage (STOP) value"***. The STOP value is identified in two stages. In stage one, the JS percentage is identified through analysis of two storage reduction base lines. The identified JS percentages were then subsequently tested for average query times for other JS percentages.

1) Stage 1: Identifying an estimated STOP value by analysing storage reductions:

To identify a STOP value, as the first baseline, overall storage requirements for a straight-forward SFC- based cell space indexing were obtained (i.e. non-DACMA solution). Next, the overall storage requirement in the case of using 4SA was obtained. Subsequently, the total storage requirement for different JS-based DACMA solutions obtained from a range the JS merge percentage values. The storage consumption was compared between non-DACMA and 4SA based scenarios. Following the comparison, the storage reduction for each JS-based DACMA solution was obtained. These steps were repeated for both single and multiple cluster scenarios. The achieved storage reductions for a single cluster is shown in Figure 7(a) and Figure 7(b). Figure 7(b) demonstrates three points in the storage reduction curve that were used to identify an estimated STOP value. Similarly Figure 7(c) and Figure 7(d) demonstrate the storage reduction curves for multiple cluster scenario and the three points involved in identifying the STOP value.

According to Figure 7, the application of DACMA initially resulted in an overall storage reduction to get decreased with the increase in the JS percentage compared to the storage demands of the two baselines. Thereafter, the storage reduction was remained constant. A notable observation in the storage reduction measured against the 4SA technique was that, beyond a certain percentage, the reduction dipped below zero. This indicated that beyond certain JS percentage, adoption of 4SA for aerial data management could be consider as a viable approach over DACMA. This is due to the 4SA's capability in reducing the storage cost. Hence, the JS percentage value where the storage reduction measured against the 4SA reached zero was considered as the upper-bound percentage for JS in finding the STOP value. Furthermore, a STOP value must

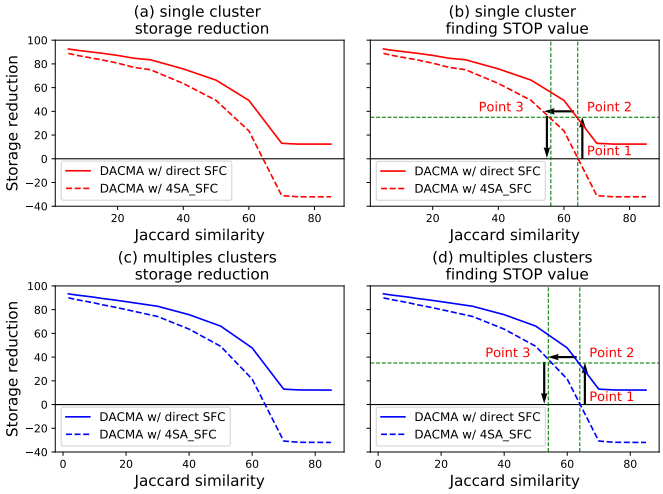


Fig. 7: Identification of estimated STOP values

yield a reasonably good storage reduction in terms of bot straight-forward and 4SA optimized SFC approaches. Thus, t methodically explore a STOP value in both storage reductic curves, three points in each curve were considered.

As for the first point (i.e. "*Point 1*"), it was decided to dra a vertical line across the JS percentage value where storag reduction becomes zero for the 4SA optimized SFC indexin The point where this vertical line intersected the DACM applied SFC based curve considered to select the secon point (i.e. *Point 2*). Therefore, a horizontal line was draw across the *Point 2*. This performed because, when navigatin to the left over this horizontal line demonstrated JS percentag values that had net positive storage reduction for both storag reduction curves.

Navigating to lower JS percentages yielded improved storage reductions. However, this was curtailed at the point where the drawn horizontal line at *Point 2* intersected the 4SA optimized SFC storage reduction curve. This was due to two reasons: (i) the overall storage reduction from DACMA was considerably net positive compared to both storage reduction curves, and (ii) to avoid potentially higher query response times. This point is considered as the third point (i.e. *Point 3*). Finally, a vertical line across the *Point 3* was drawn. The JS percentage value where this vertical line intersects the x-axis is considered as a *reasonably* good JS merge percentage. Due to the fact that at this JS merge percentage yielded net positive storage reductions for both storage reduction curves, the JS merged percentage given by the line drawn at *Point 3* was considered as the estimated STOP value.

Application of the aforementioned approach on the two storage reduction curves yielded two estimated STOP values. The estimated STOP value for the single cluster data set was 57%. For the multiple cluster data set, this was 53%. Theoretically, such estimated STOP values *must* yielded better trade-off between storage reduction and query times. Thus, further experiments were needed to obtain average query times for five different regions in the two synthetic data sets for different JS merge percentage values, along with the potential STOP values, as described in the next section.

2) Stage 2: Investigate average query response times for JS percentages identified in Stage 1:

For each synthetic data set, the average query time for five different regions were obtained. Additionally, the average query response times for the straight-forward adoption of SFC based grid cell space indexing (i.e. non-DACMA) was also investigated. To obtain statistically robust average query response times, each query was executed 25 times. The obtained average query response times a single cluster is presented in Figure 8. Due to space constraints, the average query times obtained for multiple clusters are not presented. Nevertheless, the patterns observed for multiple clusters scenario for five regions was identical to the single cluster scenario.

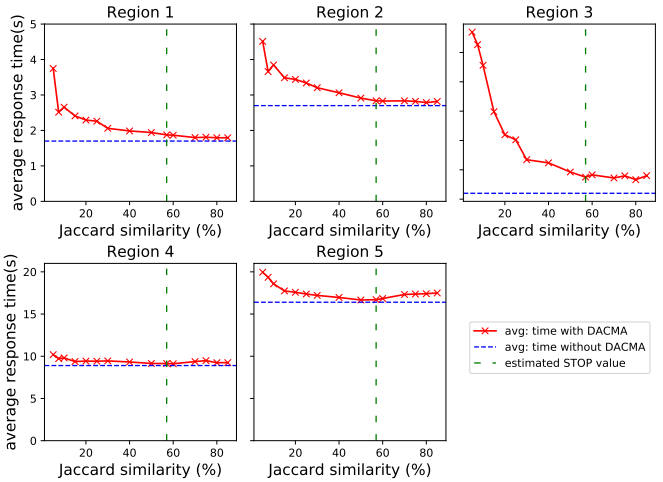


Fig. 8: Single cluster - decrease in average query response time with the increase in JS percentage

In Figure 8 the dashed blue color horizontal lines represents the average query times in the non-DACMA scenarios. The green color dashed vertical lines represents the estimated JS merge percentages identified for the the single cluster scenario- i.e. 57%. Careful investigation of Figure 8 indicates two vital observations (i) the estimated STOP value which is 57%, does *not always* yield the minimum query times, and (2) the potential STOP value, has a *significant* capability to deliver comparable average query times compared to non-DACMA query times while *always* guarantees a substantial storage reduction.

To elaborate, for the single cluster scenario, when the STOP value was 57%, the *precise* storage reductions were 56.0% and 33.7% when compared with non-DACMA and 4SA optimized techniques, respectively. (as Figure 7 was not based on values at 57% JS percentage, the respective curves do not show *precise* reductions). In achieving these reductions, the average query times for the five regions only required compromising by 0.2, 0.1, 0.3, 0.2, and 0.3 seconds. This indicated that the trade between query times at the 57% merge percentage was low. Thus, the estimated STOP value at JS 57% could still qualify as a good merge percentage.

Figure 9 demonstrates the combined observations of *Stage 1* and *Stage 2*- i.e, the impact of query response time with respect to the storage reductions when employing JS for DACMA. The green color dashed vertical lines in Figure 9 represents the storage reduction, which was 56%, at a 57% JS similarity – the estimated STOP value. The blue color dashed horizontal line represents the average query times in non-DACMA scenarios for a single cluster. Figure 9 shows that for all five tested regions, when the storage reductions reached beyond the 56% yield [the estimated STOP value (i.e. 57%)], the query times started to increase exponentially. Thus, the estimated STOP values obtained in Stage 1 and Stage 2 certainly qualify as a STOP values

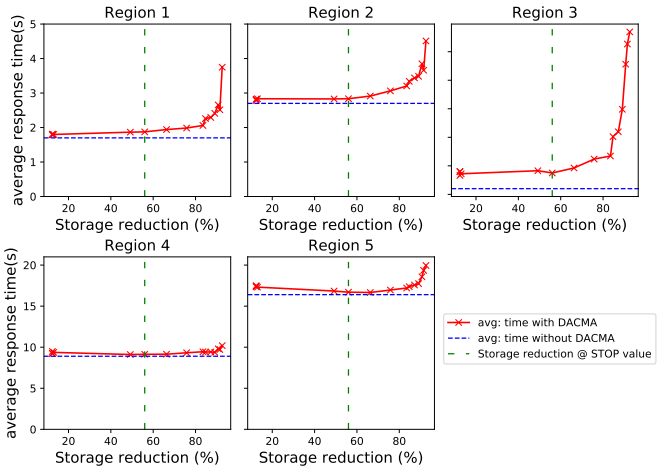


Fig. 9: Single cluster -increase in average query time with respect to improvement in storage reduction

V. Conclusions

This paper introduced a SFC-based cell space index optimization technique name as DACMA for scalable management of aerial images. DACMA's was intended to establish an adjacent cell merge algorithm based on the selection of a proper merge metric and and proper merge percentage. This paper experimentally identified that as a merge metric, JS yielded better smaller space splits, as well as improved query times when compared to PO based results. After identifying JS as a better merge metric, this work explored avenues to identify a proper merge percentage. This merge percentage had a significant impact on both storage reduction as well as query time. Thus, the focus was narrowed to identify a merge percentage that yielded substantial storage reduction, while having minimal impact on the query time. As this identified merge percentage value correspond to "space-time trade-off optimization percentage value", it was termed as "STOP value".

A. Future work

Albeit not presented due to the space constraints, this work identified a strong correlation of 0.999 between the total number of blocks and the total storage, with respect to the increase in the JS. Thus, it is hypothesized that identification of a JS merge percentage region, termed as "STOP region" for DACMA is possible. Such a region will enable selection of a range of JS merge percentages that yield efficient space-time trade-off. Therefore, the pivotal future work of DACMA is to extrapolate the determination of a STOP region using total number of blocks as a proxy.

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

[1] A. Habib, "Integration of LiDAR and Photogrammetric Data: Triangulation and Orthorectification," in *Topographic Laser Ranging and Scanning Principles and Processing*, 2nd ed., J. Shan and C. Toth, Eds., 2018, ch. 13, pp. 413–442.

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

[3] X. Zhou, X. Wang, Y. Zhou, Q. Lin, J. Zhao, and X. Meng, "Rsims: Large-scale heterogeneous remote sensing images management system," *Remote Sensing*, vol. 13, no. 9, p. 1815, 2021.

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

[5] S. Papadopoulos, K. Datta, S. Madden, and T. Mattson, "The tiledb array data storage manager," *Proceedings of the VLDB Endowment*, vol. 10, no. 4, pp. 349–360, 2016.

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

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

[8] D. Laefer, S. Abuwarda, A. Vo, L. Truong-Hong, and H. Gharibi, "2015 Aerial Laser and Photogrammetry Survey of Dublin City Collection Record," 2017, (Last accessed by 20/10/2019). [Online]. Available: https://geo.nyu.edu/catalog/nyu_2451_38684

[9] H. Samet, *Foundations of multidimensional and metric data structures*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann, 2006.

[10] C. Böhm, G. Klump, and H.-P. Kriegel, "XZ-Ordering: A space-filling curve for objects with spatial extension," in *International Symposium on Spatial Databases*. Springer, 1999, pp. 75–90.

[11] C. Hewage, A. Vo, M. Bertolotto, N. Le Khac, and D. Laefer, "4SA: Optimizing space filling curve based grid cell indexing to scalably manage remotely sensed images in key-value databases," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2022.

[12] M. Athanassoulis, M. S. Kester, L. M. Maas, R. Stoica, S. Idreos, A. Ailamaki, and M. Callaghan, "Designing access methods: The rum conjecture." in *EDBT*, vol. 2016, 2016, pp. 461–466.