Description of STARE

by

Kwo-Sen Kuo and Michael Lee Rilee

Bayesics, LLC

Bowie, Maryland

USA

Technical Memorandum 2021-001

November 2021

Abstract

We describe the spatial and temporal indexing schemes of the SpatioTemporal Adaptive-Resolution Encoding, STARE, its logical extensions, the status of its application programming interface, and examples of its applications. STARE indices are integers embedded with spatiotemporal attributes key to efficient spatiotemporal analysis. As a more computationally efficient alternative to conventional floating-point spatiotemporal references, STARE indices apply uniformly to all spatiotemporal data regardless of their geometric layouts. Through this unified reference, STARE harmonizes diverse data in their native states to enable integrative analysis without requiring homogenization of the data by interpolating them to a common grid first. In addition, the STARE scheme is hierarchical, which lends well to organizing STARE-indexed data for spatiotemporal co-alignment. Since spatiotemporal data analyses often require spatiotemporal coincidence, i.e., analyzing diverse data for the same time and space (location), spatiotemporally co-aligned data organization better guarantees scalability for distributed parallel processing, thus addressing the Big-Data challenges of volume, variety, and velocity. Moreover, since STARE indices carry not only references to spatiotemporal coordinates but also references to flexible, hence adaptive, spatiotemporal neighborhoods (intervals), it naturally generalizes to using collections of these indices for specifying "covers" of spatiotemporal extents at different resolutions. Any entity with time-dependent change of spatial extent can thus be described or specified with a collection of STARE indices. Phenomenon episodes, such as tropical cyclones, atmospheric rivers, etc., are such entities with dynamic, temporally varying spatial extents. When these STARE-index collections are appropriately catalogued in database management systems, queries to the database can easily extract the various spatiotemporal relations between the entities to enrich our understanding of their interactions.

Motivation

With the ever-increasing data volume and escalating demand for timely decision-making, parallel processing is the only means for data analysis to yield actionable information in time, in increasingly many cases. “Distributed parallel processing” is especially suited for analyses needing to concurrently process large volumes of data, with a computer-memory demand that far exceeds what is available on a single computer.

With distributed parallel processing, a group of computers, called “nodes”, connected in a high-bandwidth network, are aggregated together into a “cluster”, offering multiple times (proportional to the number nodes) the memory available on a single computer. The volumes of data to be analyzed are then partitioned and distributed to nodes in the cluster, each of which processes only a small portion of the whole and, hopefully, finishes the entire analysis collectively in a fraction of the time taken by a single computer. Clearly, the ideal scenario is one in which the pieces of data, called *chunks*, distributed to any cluster nodes are independent of (aka, uncoupled from) the other chunks distributed to all the other nodes. Each node can thus process its chunk(s) in isolation and results are only communicated to the head-node and gathered at the end when every node is done.

The real situation, however, is rarely ideal, especially with typical integrative analysis scenarios involving diverse varieties of geo-spatiotemporal data. The illustrative example in Figure 1 demonstrates this problem, namely that diverse datasets are essentially *never* aligned for efficient, scalable use of distributed, parallel computing and storage resources. The black-outlined rectangle is our region of interest (ROI), representing, for example, a grid cell of a simulation model. The ROI overlaps with, say, concurrent brightness temperature (T_B) observations from two orbiting spacecrafts. The two orbital swath segments, A and B, composed of elliptical instantaneous fields of view, IFOVs, of different sizes/resolutions, which are typically cast in array data structures in HDF files. The array indices, i.e., *i* and *j* for A and *p* and *q* for B, provide a convenient means (e.g., fixed strides) to partition the observations in A and B into chunks, as indicated by dotted lines in the figure. These chunks are distributed to a 3-node cluster using a randomized sequence, with the red numbers denoting the node to which a chunk is distributed.

However, if we wish to integratively analyze and compare, in parallel, the T_B values in the ROI from the two swath segments, the original chunks must first be repartitioned and redistributed in computer memory at compute time. For example, if we choose the partitions of B as a template for distribution, as shown in the rightmost element of Figure 1, we notice that the chunk placements of data in A on cluster nodes are not aligned with those in B! We now must move the misaligned chunk data in A to the same nodes as those in B to restore alignment before processing commences. Therefore, this misalignment of “data placements” between datasets caused by data variety induces considerable data movements between cluster nodes, detrimentally impacts

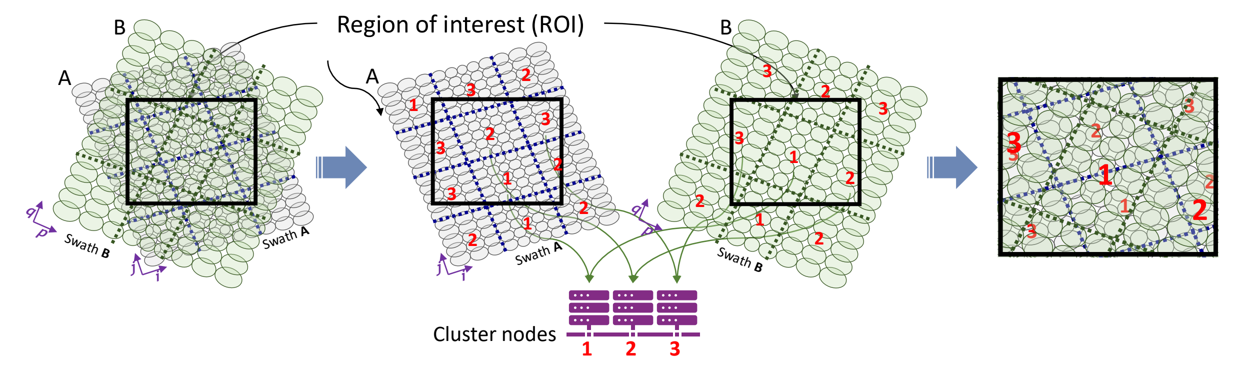

Figure 1. Illustration of data placement misalignment when using array indices.

computation efficiency and scalability [1], and thus drives up computing cost. As we can see, it is impossible to align data placements with incompatible resolutions and/or layout geometries by using array indices for partitioning. This problem is rooted in the disconnect between array indices and corresponding geo-spatiotemporal locations.

Currently, the prevailing approach to address this problem is to interpolate the diverse data to a common grid. However, this approach has at least two shortcomings. First, this computationally costly interpolation generates more data. Second, different applications may use different grid systems as the "common grid" and often have different interpolation requirements, such as whether flux conservation should be enforced and to what degree, leading to more interpolated data being generated. Moreover, the process tends to obfuscate provenance, introduces error, and endangers reproducibility.

STARE Indexing

The SpatioTemporal Adaptive-Resolution Encoding, STARE, is a geo-spatiotemporal encoding developed to support the combination of diverse data for integrative analysis [11-14**]**. STARE encodes spatiotemporal *neighborhoods* with two (2) 64-bit integers, in a left-justified manner, that contain *location* information. This mapping of space-time volumes into integers provides an excellent way to uniformly index and coordinate data with different or even irregular geo-spatiotemporal layouts in a computationally efficient manner.

Geospatial Indexing – STARE HTM

The spatial component of STARE, a modified implementation of *Hierarchical Triangular Mesh* (HTM) [15-16], encodes solid angle and is essentially a serial index into a recursive quadtree tessellation of the 8 facets of a root octahedron projected onto the sphere (see the left side of Figure 2). In other words, the STARE HTM encodes, in a 64-bit integer, a hierarchy of 8 branches (at the root level, or level 0) of quadtrees (see the right side of Figure 2). Each subsequent branching into 4 is called a *quadfurcation*. A spherical triangle of the tessellation, in any quadfurcation level (QL), is called a *trixel*. As such, a trixel in the *i*-th QL (i.e., QL*i*) is called a "QL*i trixel*"; for example,

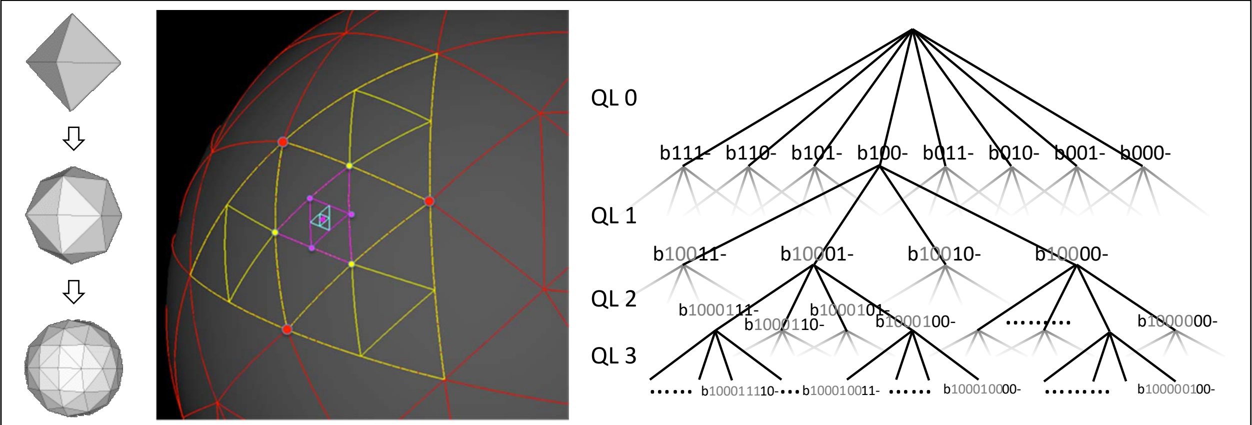

Figure 2. STARE spatial indexing scheme, based on the quadfurcation from the 8 facets of the root octahedron. (Note: our implementation has tilted the octahedron, to minimize the possibility for ambiguity in the indices todata elements located exactly at, for example, the equator, the poles, or meridians that coincide with edges of the trixels.)

a "QL2 trixel" refers to a spherical triangle of the tessellation at the 2nd quadfurcation level (i.e., the trixels of the 3rd tessellation in the left side of Figure 2).

In the following, we explain the STARE encoding of *geolocation* first and then the encoding of *neighborhood*, which by rough analogy may be thought of as a "postal address" and a "zip code," respectively. The current implementation of STARE limits the quadfurcation to the 27th level and uses the QL27 indices (akin to addresses) to index geolocation, as an alternative to floating-point

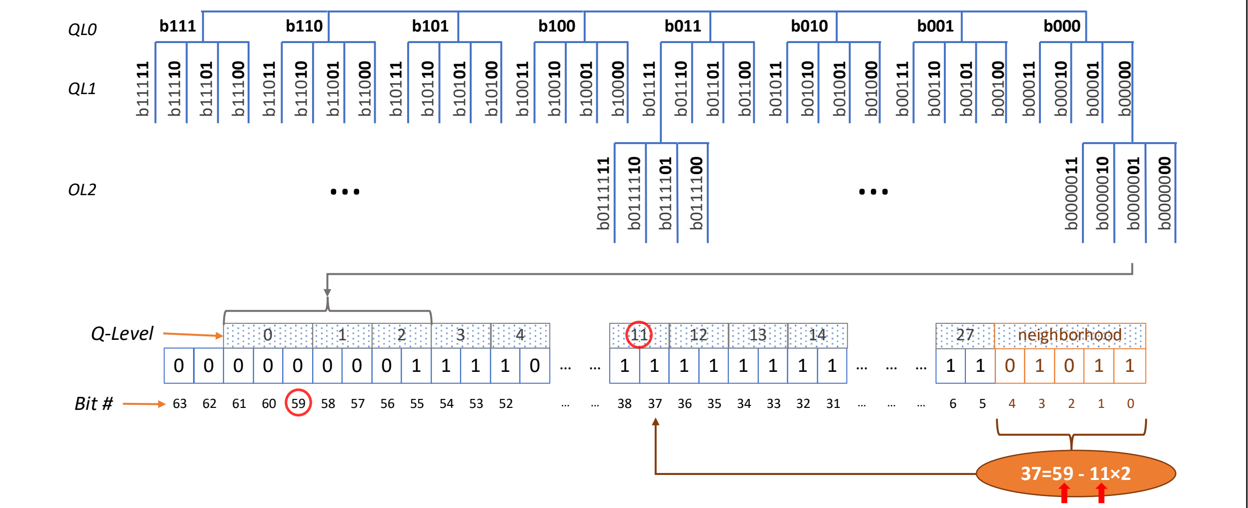
QL0
b111
b110
b101
b100
b011
b010
b001
b000
QL1
b11111 b11110 b11101 b11100 b11011 b11010 b11001 b11000 b10111 b10110 b10101 b10100 b10011 b10010 b10001 b10000 b01111 b01110 b01101 b01100 b01011 b01010 b01001 b01000 b00111 b00110 b00101 b00100 b00011 b00010 b00001 b00000
QL2
b0111111 b0111110 b0111101 b0111100
b0000011 b0000010 b0000001 b0000000
Q-Level
0 1 2 3 4
11 12 13 14
27 neighborhood
0 0 0 0 0 0 0 0 1 1 1 1 0
1 1 1 1 1 1 1 1
1 1 0 1 0 1 1
Bit #
63 62 61 60 59 58 57 56 55 54 53 52
38 37 36 35 34 33 32 31
6 5 4 3 2 1 0
37=59 - 11×2

Figure 3. Bit-pattern design for STARE spatial HTM 64-bit integer indices.

longitude-latitude (lon-lat) pairs. At QL27, the edges of the trixels have lengths of ~7-10 cm, leading to a rather uniform ~7-10 cm geolocation uncertainty over the globe, which is generally better than the commonly used single-precision floating-point lon-lat pairs, especially near the equator. As it can be seen, 8 (=2^3) branches of 27-level quadtrees (=$2^{2\times 27}$=2^{54}) take 57 (3+54) bits to encode and there are 7 bits left in the 64-bit word, of which we use the least significant 5 bits (akin to zipcode) to indicate neighborhood.

Figure 3 illustrates the scheme with QLs 0-2. The bit pattern of a STARE HTM 64-bit-integer index is left justified at bit #61, the 3rd most significant bit position. Three bits, i.e., #59-61, contain the binary number that indexes one of the 8 root octahedron facets. The next 2 bits, i.e., #57-58, contain the binary number that indexes one of the four trixels of the 1st quadfurcation, and so on until QL27, which we use to index the center of a data element, e.g., either a point observation, an instantaneous field of view (IFOV) from a satellite-borne instrument, or a model simulation grid cell. If the areal coverage extent of the data element is ~16 km² (=4-km×4-km), equivalent to a nominal resolution of 4-km that is smaller than the ~5-km edge length of a QL11 trixel but greater than the ~2.5-km of a QL12 trixel, we say that the data element has a *trixel neighborhood* of QL11 and set the neighborhood bits, i.e., #0-4, to the binary equivalent of 11.

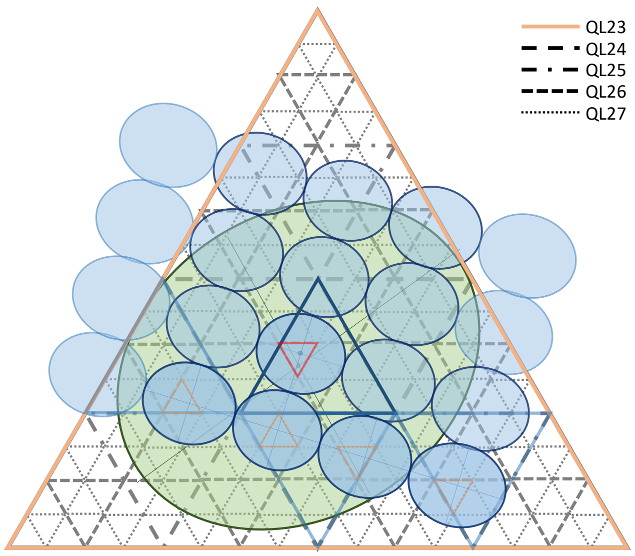

Figure 4. STARE HTM indexes locations and neighborhoods.

A geospatial data element nearly always has an areal extent associated with it, like a zip code in our postal analogy mentioned above. It is a key piece of information required for most geospatial analyses but missing from the geolocation-only reference of the lon-lat pair. STARE, however, embeds that information in its indices and leverages it for harmonizing geospatiotemporal data with diverse resolutions and layouts.

Note that, to assert whether a data element x overlaps with or contains another data element y, in terms of mathematical set logic, we examine first the respective neighborhood bits (zip codes) of their STARE HTM indices and choose the smaller number. For example, if the neighborhood bits of x have a value of 11 and y a value of 13, we choose 11. We then examine values of both indices in the range of bits corresponding to QL11 to QL0, i.e., bits #37-61 (which can be performed efficiently with bit masking). If they are the same, then x contains y, because it means y is an offspring of x in the STARE HTM tree hierarchy and an ancestor trixel (lower numbered QL) contains an offspring trixel (higher numbered QL).

Figure 4 illustrates these ideas graphically in detail. It shows a QL23 trixel outlined in peach color, containing a mesh of sub-trixels (offspring on the quadtree) from QL24 to QL27. A large IFOV is shaded in green. The centroid of this green IFOV falls in the red-outlined QL27 trixel whose STARE HTM index value is thus used for indexing the IFOV's location. Since the area extent of the green IFOV is larger than a QL24 trixel and smaller than a QL23 one, and the peach-color-outlined QL23 trixel is the ancestor on the quadtree of its QL27 location, the peach-colored trixel is used to represent the IFOV's neighborhood.

A group of 20 (=5x4) smaller IFOVs in Figure 4, from a different instrument, is shaded in blue, whose area extents are larger than a QL26 trixel but smaller than a QL25 one. Thus, the coarser QL25 trixels are used to represent their neighborhoods. The centroid of one of the blue IFOVs also falls into the red-outlined QL27 trixel. This blue IFOV therefore has the same STARE location index value as the larger green IFOV but it has a QL25 trixel (outlined in darker blue) neighborhood. A subset of these 20 blue IFOVs are outlined with a darker shade of blue to denote that their QL27 locations (and corresponding QL25 neighborhoods) are contained by the peach-outlined QL23 neighborhood of the green IFOV.

It is thus apparent that, with such a scheme, we can carry out efficient set operations, e.g., intersect and union, using integer STARE indices to yield fairly accurate approximate results. This type of set logic operations is generalized in the STARE application programming interface (STARE API, described below) to collections of STARE trixels with integer-range operations.

Temporal Indexing – STARE HCP

The temporal component of STARE, i.e., the *Hierarchical Calendrical Partitioning* (HCP), has a similar structure, except that different levels of the tree, corresponding to different calendrical units, i.e. hours, days, months, have different numbers of branches. The HCP is based on a hierarchical partitioning of International Atomic Time (TAI) and API functions are provided for both UTM and TAI, with low-level support provided by the Essential Routines for Fundamental Astronomy (ERFA) derived from the International Astronomical Union's Standards of Fundamental Astronomy (IAU/SOFA). HCP is designed to support several scalings, i.e., different calendars/resolutions, see Table 1, for an example. STARE HCP indices also include asymmetric neighborhood information. For example, in addition to the time of an observation, rough measures of the

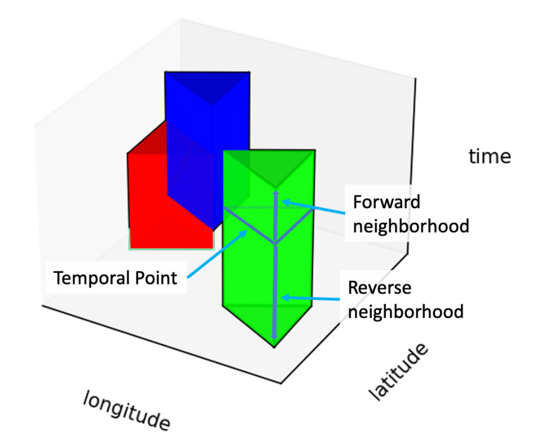

Figure 5. STARE spatiotemporal cells. Each prism corresponds to two 64-bit integers.

Table 1. Example configuration of STARE HCP temporal encoding.

Meaning	No. Bits
Calendar Type	2
Reverse neighborhood	6
Forward neighborhood	6
millisecond	10
Second	6
Minute	6
Hour	5
Day	3
Week	2
Month	4
Year	13
Before/After	1
Total	64

time since a prior observation and the time to a future observation are encoded in the same 64-bit index value. With this encoding, temporal set logic can be performed in a manner similar to STARE HTM indices.

Figure 5 shows visual representations of the fundamental *spatiotemporal volume element*, with each "prism" being associated with two 64-bit integer, i.e., HTM+HCP, indices. The green spatiotemporal prismatic volume element in Figure 5 shows the asymmetric temporal neighborhoods, typical for observations from instruments on low Earth orbit (LEO) satellites, because their revisit periods are not uniform, depending not only on latitude but also cross-track position in the swath. Complex spatiotemporal regions of interest can be represented as sets of STARE HTM and/or HCP indices, leading to the concept of a STARE cover described below.

STARE Covers

As alluded to above, a STARE cover is an area or a volume, represented by a set of STARE HTM and/or HCP indices, "covering" the spatial or spatiotemporal extent of interest, for example, the temporal evolution of a storm's area. That is, if we use STARE spatial (HTM) and temporal (HCP) index tuple (s_i, t_i) to denote the spatiotemporal prismatic volume elements in the covering set, then $V \subset (C_V = \cup_i (s_i, t_i))$, where V is the spatiotemporal volume of interest and C_V is the cover of V. The geo-spatiotemporal semantics of the integer indices allows set logic operations to be performed through integer operations and tree-based algorithms, usually obviating floating-point geometric calculations for STARE-indexed data.

Spatial Covers

A time slice (or a snapshot) of a spatiotemporal volume, e.g., a storm area at a given time, or a region of interest (ROI) that stays essentially constant with time, e.g., state of Oklahoma, needs only a spatial cover, i.e., a collection of STARE HTM indices. Figure 6 shows an example spatial cover for Australia and Tasmania constructed using STAREPandas, a part of the STARE API and a pendant to GeoPandas.

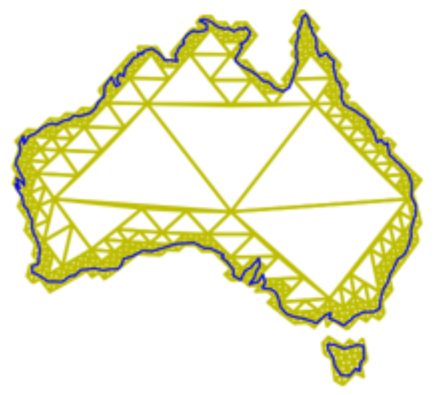
Figure 6. STARE spatial cover for Australia, including Tasmania.

The variety of trixel sizes in the cover correspond to the variety of embedded neighborhood or resolution information (QL) of the STARE HTM indices. The user specifies the maximum QL (i.e., finest neighborhood, thus highest resolution) trixels to use in a cover. The trixels of highest QL usually concentrate around the border of the region of interest. The higher the QL trixels used, the more accurately the cover approximates the ROI, but with increasing number of total trixel indices required for the cover, which increases storage/memory footprint and computation resource demands.

As a set of 64-bit integers corresponding to areas, no special treatment is required to handle the Tasmanian strait, those trixels are simply not included in the cover. In other words, the collection of STARE indices in a STARE cover is a very general way of specifying region(s) or volume(s) of interest. STARE HTM indices treat topological varieties, such as fragmentation, holes (e.g., lakes) in the region, or bubbles in the volume, consistently without partiality.

Spatiotemporal Covers and Moving Objects

Many investigations in Earth Science have a phenomenon focus, for example the investigations of tropical cyclones, atmospheric rivers, El Niño, etc. To capture the temporal evolution of a phenomenon episode, e.g., Hurricane Irma, a three-dimensional (3D) spatiotemporal STARE cover, i.e., 2D in space plus 1D in time, is needed. As alluded to in Figure 5, a STARE spatiotemporal volume cover is a collection of (triangular) prismatic volumes.

Figure 7 shows two hypothetical phenomenon episodes, one moving mostly northward while the other mostly eastward. Plot A in Figure 7 shows the spatial cover of each time slice in varying colors, whereas plot B shows the projection of those time slices on the 2D lon-lat plane. Taking the union of the time slices of STARE spatial covers yields spatiotemporal covers for these two episodes. When moving objects, such as phenomenon episodes, are expressed with STARE HTM+HCP indices tied to space-time locations and neighborhoods, they become amenable to numerical operations and manipulation, such as subsetting and differencing, and support database ingest, cataloging, and queries, as well as scalable, parallel computation.

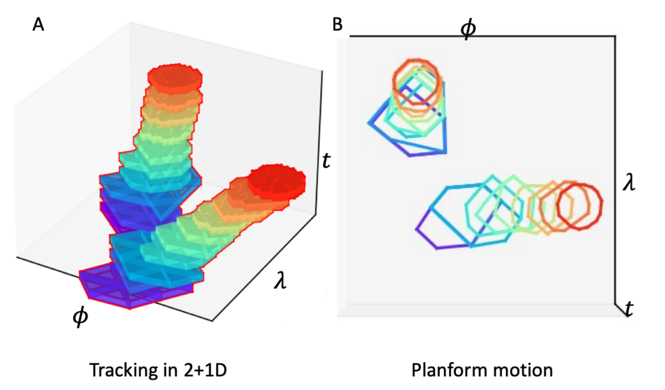

Figure 7. Two moving objects represented with STARE spatiotemporal covers in 3-D (left) and planform (right) renderings.

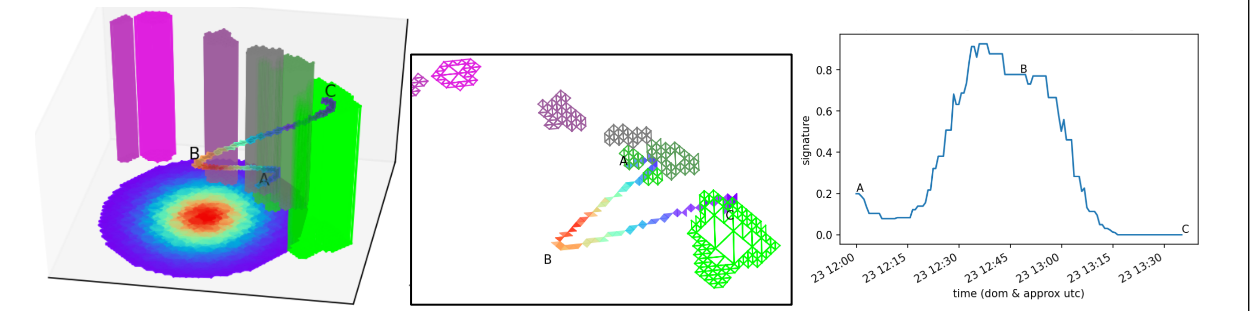

Figure 8. A notional scenario where an aircraft follows a trajectory (ABC) over some phenomenon (rainbow disk, left) making observations indicated along the trajectory (middle) and as a time series (right). STARE covers of the Hawaiian Islands are shown as columns.

In Figure 8, the moving object is an instrumented aircraft, sent from Lanai Island (point A) to observe, by remote sensing means, some (hypothetical) phenomenon episode over the ocean in the west-south-west direction. After reaching the sky near the center of the phenomenon (point B), the aircraft returns to Hawaii Island (point C). Since the islands are stationary, their spatial covers do not change with time. Therefore, they look like vertical columns in the leftmost plot. (We could have shown the temporal evolution of the episode along the vertical time axis as well, but it would have obscured the flight path.) The projection of the aircraft's time-dependent location and the neighborhood of its measurements is depicted in the center plot. The rightmost plot shows the value of the variable measured by the aircraft as a function of time (and location).

STARE Data Harmonization

STARE harmonizes geo-spatiotemporal data varieties for fusional analysis in their native resolutions and layouts. This data-variety harmonization facilitates data placement alignment first in *storage* for efficient search-and-filter and second in *memory* for performant and scalable distributed parallel processing. It thus supports our STARE application programming interface (API) to enable timely, custom, and on-demand fusional analyses, minimizing the need for costly wholesale homogenization through interpolation/re-gridding that often obfuscates the processing provenance and generates massive custom intermediate data of limited reusability to other users and communities (see Figure 9).

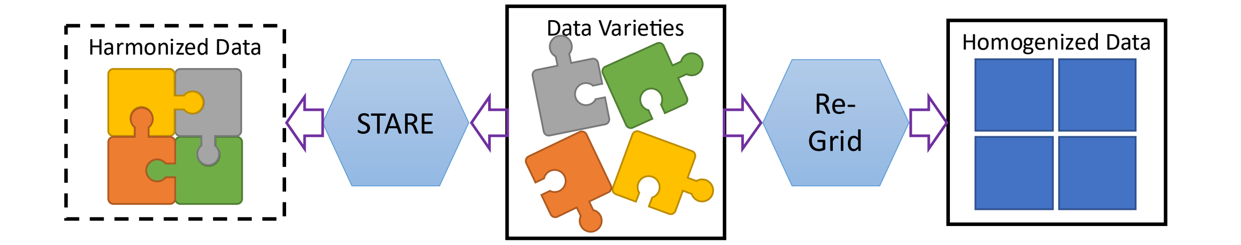

Figure 9. STARE Data harmonization avoids the creation and proliferation of excess intermediate data.

STARE-PODS – Variety-Harmonized Organization

Our STARE Parallel Optimized Data Store, STARE-PODS, leverages the STARE hierarchy to facilitate performant distributed parallelization even further, by fully exploiting STARE hierarchical indexing for data packaging and organization.

Take, for example, a swath segment and partition it into chunks at a prescribed quadfurcation level, QL*q*, as shown in Figure 10. It naturally suggests a hierarchical organization as illustrated in Figure 11. On traditional filesystems, one can implement a directory structure to mimic the hierarchy. Each directory corresponding to a QL*q* trixel is termed a *STARE pod* or, simply, a *pod*. Chunks, from all datasets, belonging to the same pod reside in the corresponding directory.

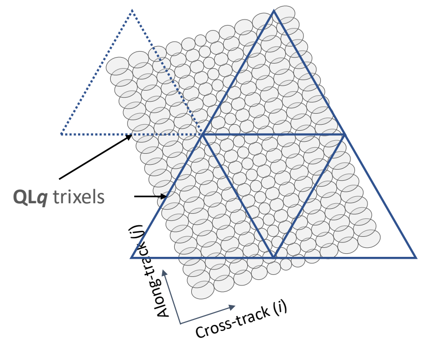

Figure 10. Partitioning a swath segment at QLq.

For Web Object Stores favored by Cloud vendors where directory structure is absent, one could prepend the hexadecimal number of the QL*q* trixel index to the name of the chunk. For example, if we have decided to partition at QL4, i.e., $q = 4$ (requiring 11 bits and 3 digits in hexadecimal), and the original swath segment has “XYZ” as its file name, we could prepend the names of the 4 chunks in Figure 11 with hexadecimal number corresponding to bits #51-62[1] (recalling Figure 3) of the QL4 trixel HTM index, resulting in, e.g., “0x6FC_XYZ”, “0x6FD_XYZ”, “0x6FE_XYZ”, and “0x6FF_XYZ”, etc. One may just as easily use these filename prefixes as a directory structure for search and filter to align data placements. Or,

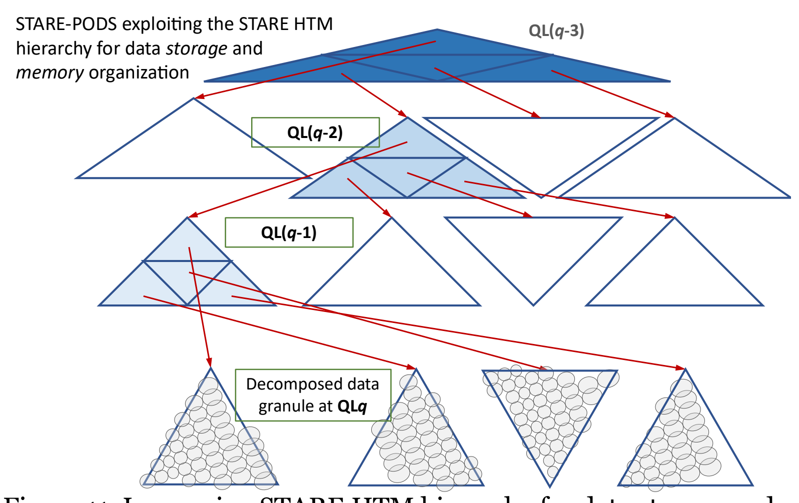
STARE-PODS exploiting the STARE HTM hierarchy for data *storage* and *memory* organization
QL(*q*-3)
QL(*q*-2)
QL(*q*-1)
Decomposed data granule at QL*q*

Figure 11. Leveraging STARE HTM hierarchy for data storage and memory organization to facilitate scalable spatiotemporal data alignment.

[1] There are 12 bits in bits #51-62, i.e., one bit more than the required 11 bits for indexing QL4 trixels. This is because, for consistency’s sake, we keep bits #59-62 (4 bits) as a group. This is especially advantageous when expressing the indices as hexadecimal numbers since each hexadecimal digit represents 4 bits.

perhaps even better, one could employ a database management system (DBMS) to catalog and organized the chunks according to the STARE hierarchical indices as well.

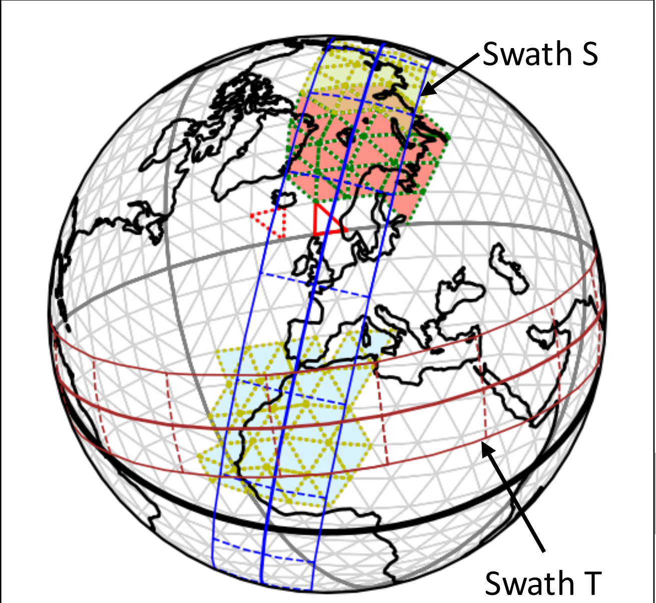

Figure 12. Illustrative example of two intersecting orbital swaths of different orbit characteristics overlaid on a mesh of STARE HTM QL4 trixels.

Figure 12 illustrates the advantage of repackaging and organizing according to STARE hierarchy using two notional orbital swaths. The thick, solid black line is the equator. The solid dark gray lines of medium thickness outline the QL0 trixels (i.e., the projection of the root octahedron's edges), whereas the thin light gray lines outline QL4 trixels. Observations from two satellites trace the two swaths: swath S for a sun-synchronous orbit (blue outline) and swath T for an inclined orbit targeting the tropics (brown outline), with the satellites' nadir tracks indicated by thicker solid blue and brown lines, respectively. These observations are broken into swath segments as indicated by the cross-track dashed lines. Currently, data elements (IFOVs) within each segment are typically cast into a raster array and packaged into a file.

To implement STARE-PODS, we would first generate STARE indices for all data elements in the files. If we decide to use QL4 trixels for partitioning the data, we then repackage data elements within each QL4 trixel (having non-empty overlap with the swaths) into a file as a STARE *chunk* (see Figure 11). Partially filled chunks, like the QL4 trixel just south of Iceland with dotted red outline (as opposed to the filled one north of Britain with solid red outline) in swath S, are permitted.

The QL4 trixels shaded in red form the *QL4-trixel cover* for one swath S segment, whereas the ones shaded in green form another for a neighbor segment (to the north). Since we wish to preserve the original swath-segment appearance, there will be two sets of partially filled chunks for the QL4 trixels (those shaded by both red and green) at the shared boundary of these two segments, one set for each segment with the data elements of the other segment excluded.

Note the chunks in the cyan shaded QL4 trixels in Figure 12 covering the overlap of swaths S and T. If we use a directory structure to implement the STARE hierarchy, they will be under the same respective QL4 directories. If we use QL4 trixel index for chunk-name prefix, corresponding chunks will have the same prefixes. Finding the overlaps between swath S and T (or any other datasets) becomes trivial and scalable! Since our STARE API can convert arbitrary (spherical) polygons into STARE covers at any given QL, finding overlaps between an ROI and any STARE chunks becomes trivial as well. With STARE-PODS, geospatial analytics will become more resource efficient, more performant, and more scalable.

We have prototyped a STARE-PODS with STARE-based data partitioning and organization on a traditional file system. This prototype, named *XCAL*, uses three 1-month of cross-calibrated microwave radiometer/imager datasets produced by the NASA Precipitation Processing System [17] for NASA Precipitation Measurement Missions, which include (cross-)calibrated brightness temperatures from 18 satellite-borne microwave radiometer/imager instruments in NASA's Global Precipitation Measurement (GPM) mission era, such as AMSR2, ATMS, GMI, MHS, SSMIS, etc. [18] This set of data products exhibits not only data varieties across the products of different instruments but, due to different IFOV resolutions for different microwave frequencies, also within the same product of the same instrument. For example, the SSMIS instruments (onboard different satellites of the Defense Meteorological Satellite Program) have 4 distinctive swaths (S1-S4) for their 8 microwave frequencies and 11 channels (including different polarizations for some frequencies) [18]. Such an acute diversity of data varieties constitutes a serious test for our STARE-PODS concept, which has turned out to be very positive. Our developer and tester have commented, “I cannot imagine the amount of effort that would have been required to conduct integrative analysis on these data without the STARE-based partitioning and organization.”

As a more concrete example from XCAL, Figure 13 shows part of a data granule from SSMIS, a microwave radiometer on a polar orbiting spacecraft, in which the IFOVs in the granule have been partitioned into chunks according to their STARE spatiotemporal location. Represented as triangles, the pods in this case are at QL4, which have a corresponding scale size of ~640 km. As illustrated in Figure 11 and Figure 12, STARE-PODS, our data store architecture, organizes chunked data from different data sources, co-aligning them to minimize data movement for storage and use on scalable resources, such as HPC and Cloud. A metadata database, or even an appropriately designed file system directory (folder) hierarchy can rapidly find co-aligned chunks from different datasets.

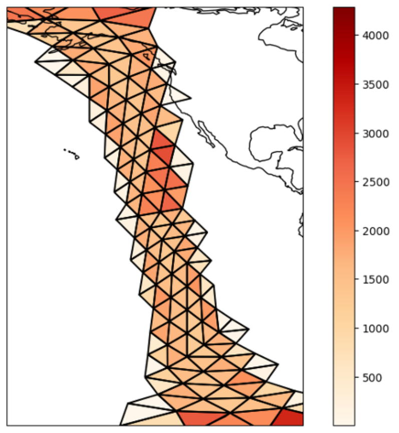

Figure 13. SSMIS IFOVs chunked according to STARE pods, with the redscale indicating the number of observations in a chunk.

STARE API

A variety of functions for working with STARE indices and collections of them are provided for C++ and Python applications, as shown in Table 2. The goal of the API is to shield users from requiring intimate knowledge of bit-level manipulations of STARE indices, so their scientific productivity can be enhanced by working at a higher level of abstraction. Functions are provided for translating between latitude-longitude (lon-lat) and for indexing and organizing data.

With PySTARE, Python users can pass around Numpy arrays of STARE indices using the API to perform rudimentary construction tasks from geolocation data, followed by (conditional) subsetting, and other spatiotemporal set logic on any diverse data that has been indexed and harmonized using STARE. Building on PySTARE, STAREPandas integrates STARE-based geo-spatiotemporal operations with the Pandas dataframe by extending GeoPandas.

Table 2. STARE Application Programming Interfaces

C++ Library API	
Spatial Indexing HTM	• Translations to/from latitude/longitude • Integer intervals & comparisons • Convex hulls and cover sets • Trixel geometric information • Rotated root polygon • Nearest neighbor and resolution • Skip-list-based region-of-interest objects
Temporal Indexing HCP	• TAI, UTC based on ERFA (SOFA) • Julian day support • Temporal interval and resolution support

PySTARE & STAREPandas
“Pythonic” interface to STARE functions via SWIG Numpy arrays Matplotlib/Cartopy Support GeoPandas integration with STARE dataframes

Using the STARE API allows researchers to construct higher-level geo-spatiotemporal objects such as regions of interest or dataset covers and, in turn, to distribute and co-align multiple datasets according to STARE, leading to harmonized, scalable data ready for further work, including physics-based detailed analyses, e.g., at the IFOV level, that are *impossible* with the current state-of-the-art interpolation-based processing, which irreparably washes away physical detail.

STARE API Application

STARE provides a uniform way to refer to spatiotemporal regions that harmonizes data on computational resources. STARE indices act as coordinates, which bring diverse data together because of STARE's combination of real-world and cyberspace data organization (Figure 14).

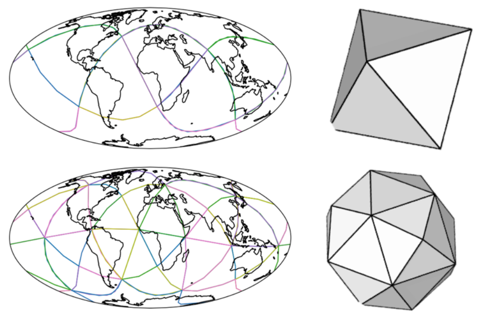

Figure 14. The QL0 (root) and QL1 polyhedrons (right) underlying the HTM neighborhoods in Mollweide projection (right). Note the "tilt" of the root octahedron.

shows the harmonization and fusion of two datasets with quite different patterns of observation. Imagery from GOES and swath data from MODIS were indexed (i.e. geo-locations plus neighborhoods encoded with STARE), partitioned, and combined. The data may be indexed in a variety of ways. The STAREmaster command line app makes sidecar files for corresponding legacy files, enabling users to access and process data in legacy files leveraging STARE. PySTARE and STAREPandas provide functions for constructing STARE indices from lon-lat pairs. Because STARE is a geo-spatiotemporal indexing scheme and does not re-grid, no information is lost during indexing or partitioning. Despite the distortion due to projection, the QL5 trixels of the GOES cover in Figure 15 do not vary greatly in size and there is no difference in the way poles and lower latitudes are handled. Once the data are indexed, partitioning and sub-setting according to geo-location become straightforward set logic based on integer operations. Specifying a region of interest (ROI), like the circle in Figure 15, allows one to select off or bin STARE-indexed data with as fine a granularity as needed (down to ~7-10cm across). Using ROI covers, which are just arrays of 64-bit integers, to select data in PySTARE is generally no more complicated than indexing using *numpy.where()*. STAREPandas operates on a higher abstraction level still.

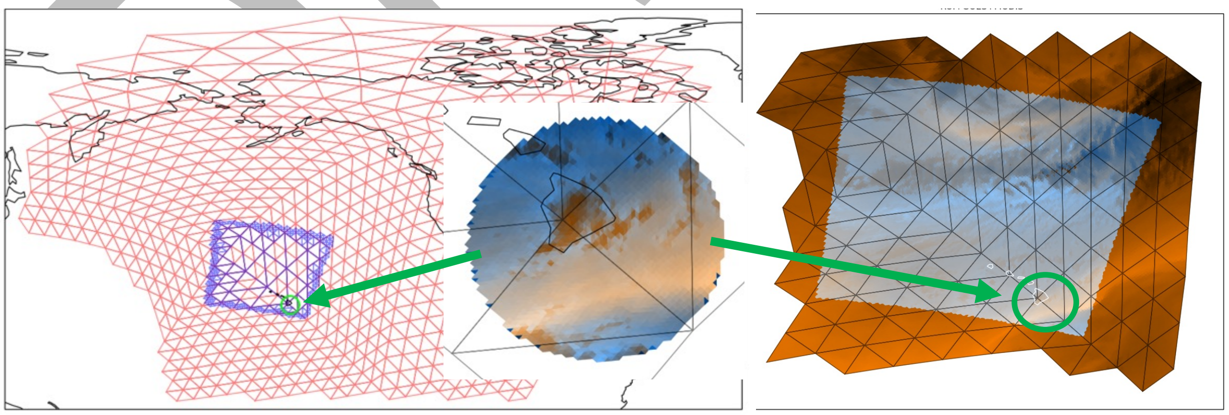

Figure 15. GOES (red/brown) and MODIS (blue) granules harmonized and integrated using STARE, visualized in an equirectangular projection. Covers for the GOES and MODIS granules are shown with fixed and variable QL for GOES and MODIS, respectively, on the left. Data may be partitioned for parallel analysis or storage by STARE trixel neighborhoods. A circular ROI is shown as an example of sub-setting these two diverse data sets.

Figure 16 shows GOES and MODIS data being harmonized using PySTARE. A circular region of interest (ROI, same as that in Figure 15) has been specified and the API has been used to construct covers using QL10 and QL11 for GOES and MODIS, respectively. A simple comparison of bit-patterns allows data from the two data sets to be overlaid, or harmonized, even though they have very different geographical layouts. (Note that the MODIS cover is smaller than the GOES cover because a finer QL, i.e., 11 versus 10, is used for MODIS cover.)

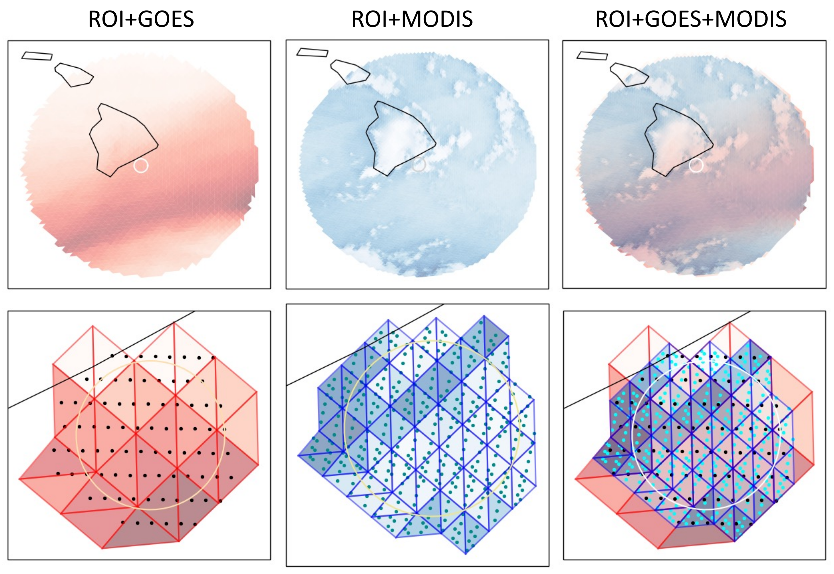

Figure 16. Covers of a circular ROI for GOES and MODIS data, showing the harmonization of data with diverse geometric layout. IFOVs are black (blue) for GOES (MODIS) and the trixels are filled with a color based on aggregating the data values within a trixel. The red HTM trixels have a length scale of approximately 10km. Note the GOES data have been selected or "joined" with the higher resolution MODIS data. The lower row shows the detail of the circle indicated in the upper row.

Both the PySTARE and STAREPandas APIs provide the means to subset data based on covers. Any spatiotemporal region can be represented by a cover, such as the geographic circle in Figure 16 or even a data-driven ROI, such as a precipitation event, Figure 17. In Figure 17, NMQ[2] radar observations of a precipitation event have been indexed with STARE. Given that NMQ is a gridded data product, it suffices to calculate the STARE indices once as the grid locations do not change. As an example of data-driven *conditional sub-setting*, a precipitation-rate threshold has been set on the NMQ data, which is used next to subset near-concurrent TRMM swath, with the result displayed as a Boolean mask seen in Figure 17 (left and center).

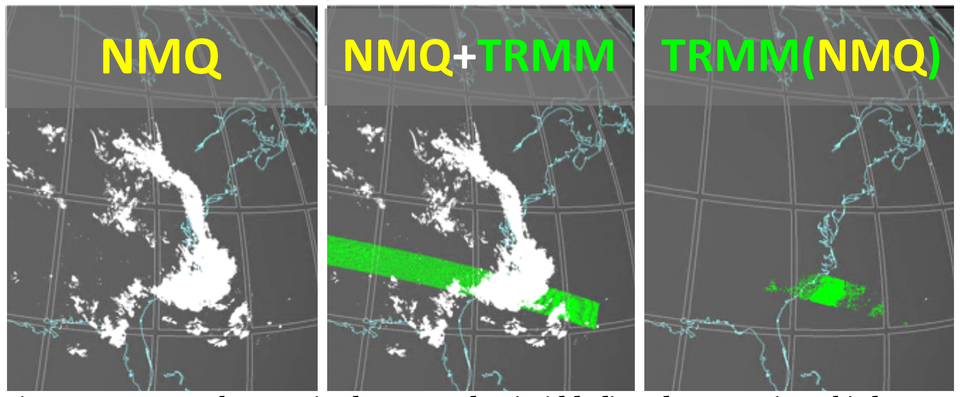

Figure 17. STARE-harmonized NMQ radar (gridded) and TRMM (swath) data used for data-driven conditional sub-setting. STARE-locations of NMQ data above a given threshold are used to select TRMM observations (right).

STARE-harmonization

[2] National Mosaic and Multi-sensor QPE (quantitative precipitation estimates) produced by NOAA National Severe Storm Laboratory (NSSL). It has been superseded by the Multi-Radar Multi-sensor System (MRMS) data product since ~2015.

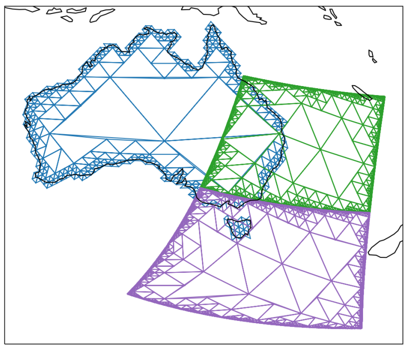
Figure 18. Shows covers associated with a GeoPandas-derived ROI (Australia and Tasmania) and two MODIS granules.

makes it easy to determine which data of the TRMM swath are in the same neighborhoods of the NMQ ROI (Figure 17, right). This subsetting process needs not stop here. The cover of the TRMM(NMQ) can be used to search, select, and subset data from other, diverse data sets, without regard to their geo-spatiotemporal layout. Just as a feature detected in the gridded NMQ data was used to select the swath TRMM data, the conditionally sub-setted TRMM(NMQ) cover can be used to load data from MODIS, VIIRS, aircraft, ground stations, and so on.

On a more macroscopic scale, Figure 18 demonstrates some of the cover and indexing functions of STARE. A cover of Australia and Tasmania is shown constructed using STAREPandas calling on underlying GeoPandas functions and data. Figure 18 also shows two covers constructed from two MODIS granule files of swath data. These data have been indexed using the STAREmaster command line application, which constructs sidecar files with translation lookup tables for mapping between STARE index values and their native array location in legacy files (i.e., as they are currently stored in files in data centers or on Cloud). The STARE sidecar files allow STARE-aware applications to use pre-computed STARE indexing to harmonize data, readily co-aligning different kinds of data without interpolation and dramatically reduced data wrangling. Covers and lookup tables can also be constructed via Python. STAREPandas is readily extensible by end users to work with diverse Earth Science data in legacy file formats or on Cloud.

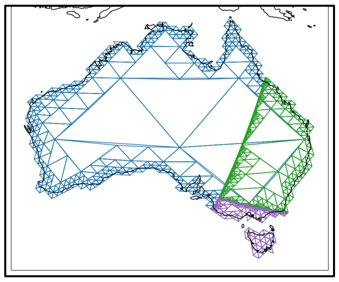
Figure 19. The MODIS granules of Figure 18 are trimmed to the ROI. STARE, PySTARE, and STAREPandas provide the capability to determine intersections of covers.

Unlike conventional raster array indices that are devoid of spatiotemporal semantics, STARE indices do double duty as spatiotemporal coordinates and "array-like" indices. Rather, key-value stores using STARE indices are almost as easy to use as array indices, even when the data have different spatiotemporal layouts.

Figure 19 shows the result of using STAREPandas to determine the intersection of the MODIS granules and the ROI. We can compare the trimmed covers, i.e. the intersections of the granule covers with the ROI, with the translation tables in the STAREmaster sidecar files to determine the data elements to load from the legacy data files, e.g., Figure 20. The data shown in Figure 20 are actually MODIS Level-2 IFOVs rendered as dots (not to scale for visibility) and have not been interpolated to any grid, highlighting the fact that STARE has been used as a harmonizing mechanism (i.e., indexing and organizing) as opposed to homogenizing by interpolation to a predetermined grid. As stated elsewhere, the Tasmanian Strait requires no special handling. The STARE indices (64-bit integers) associated with the strait are simply not in or among the indices of the ROI cover.

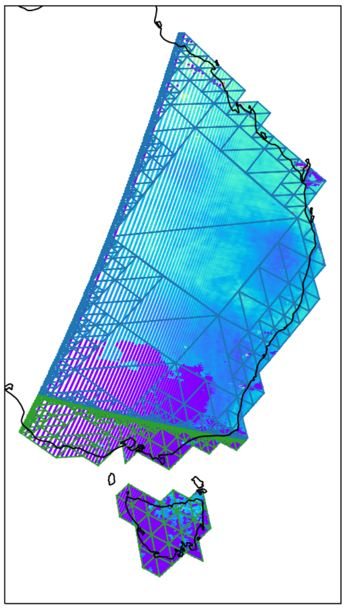
Figure 20. Using the trimmed covers to sub-set the MODIS data.

The simplicity of using integer indices with spatiotemporal semantics for searching, intersecting, and sub-setting is apparent when STARE-based catalogues are constructed. Figure 21 shows a set of MODIS granule covers that we catalogued to support further analyses, as a proof-of-concept prototype.

STAREPandas and STARELite (a SQLite extension) provide high-level support for the storage organization of geographical regions as STARE-covers and provide querying functions for determining when and where query-ROIs intersect with data in the database. The STARE covers can be generated on the fly on ingest or be precomputed, e.g., by using STAREmaster. For the case illustrated in Figure 21, the data were stored in Cloud, STAREmaster was used to construct sidecar files, which were then read into both STARELite and STAREPandas databases. In fact, STAREPandas provides a turnkey *folder2catalogue()* function for the construction of STARE indexes for harmonization from folders on local filesystems or in Cloud (e.g., AWS S3).

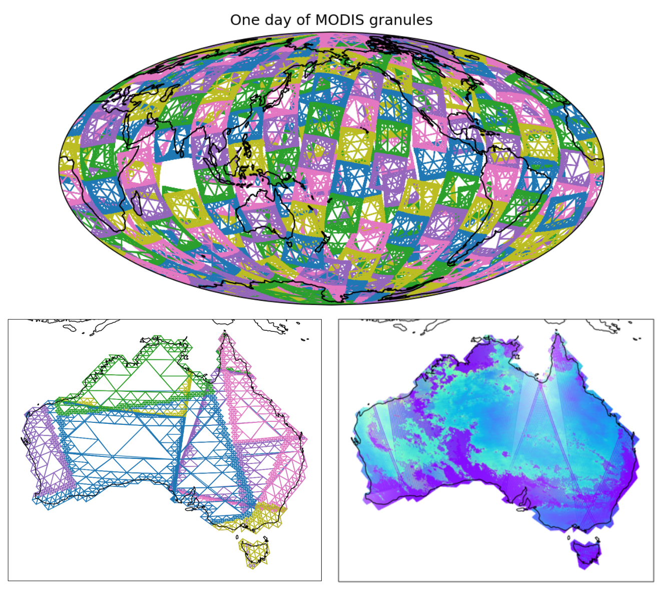

Figure 21. A catalogue of MODIS granule STARE covers can be loaded into a STARE-aware database or STAREPandas (top). A STARE cover for an ROI (Australia) can be used to query the catalogue to determine which granules overlap the ROI (left), which can then be used to load non-interpolated IFOVs that intersect the query region (right).

The results of querying a catalog with an ROI are shown in Figure 21 (left) where we see the trimmed granule covers of the catalogued MODIS data overlaid on the ROI as in Figure 19. Using a database or dataframe provides a high-level means for organizing, querying, and accessing other geo-spatiotemporal functions, but it is not difficult to use the lower-level PySTARE or C++ APIs to achieve the same effects. Performing geo-spatiotemporal logic with STARE is in many ways much simpler than working with geometrical entities such as points, lines, polygons, etc. represented as vectors of floating-point numbers. The source datafiles in Figure 21 were stored in Cloud, catalogued, and then processed in parallel. Figure 21 (right) shows the data collected from the catalogued granules matching the ROI in the query.

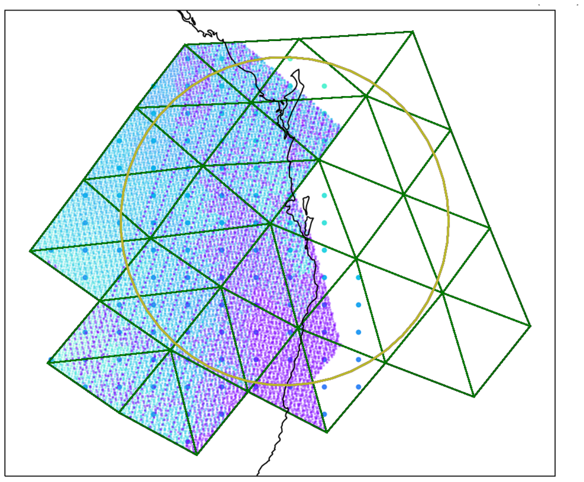

Figure 22. MERRA-2 data (gridded; green dots) and MODIS IFOVs (shades of blue) are harmonized on the Australian ROI in the vicinity of Brisbane, Australia. The green trixels are from a cover of the MERRA-2 data. Note higher resolution neighborhoods (finder trixels) of the MODIS data. The coastline visualization uses higher resolution data than that used to construct the Australian ROI cover. Simple Python dictionaries were used for this harmonization, where STARE spatial locations were used as dictionary keys.

A harmonization example is shown in Figure 22, where a subset of MODIS and MERRA-2 data matching the Australian ROI have been harmonized into QL6 pods (green trixels) in the vicinity of Brisbane, and each pod has been rendered separately. In this simple example, the pods are stored in a Python dictionary using the pods' STARE spatial location index value (a 64-bit integer) as a key. There has been no information loss in this harmonization as no interpolation or other processing of the data values has occurred. The data have merely been indexed and organized and become now *analysis-ready*.

The partitioning according to STARE greatly simplifies further processing using parallel or elastic computing resources. Researchers wishing to interpolate or re-grid the data are free to do so when necessitated by the analysis and can take advantage of the partitioning for scaling up volume and variety. Since the data are still in their original form, STARE enables the work of researchers wishing to apply physics-based analyses involving the observational details and context at scale, e.g., globally.

As a simple example of a parallel, distributed analysis performed on harmonized data, Figure 23 shows a basic statistical analysis of water vapor observations contained in the catalogued MODIS data (one day's worth) in Figure 21. In this case, we have chosen QL3 pods as the ROIs to make the results more visible. As the analysis is identical for each pod, the analysis of the whole is pleasingly parallel. The work associated with searching for and loading the data and then calculating summary statistics for each pod was executed in parallel using the Daskhub parallel environment in NASA's Science Managed Cloud Environment (SMCE).

For each of the 512 QL3 pods, the catalog was searched for overlapping MODIS Level-2 data. STARE's catalogue searching is efficient as it compares ROI-cover to the granule-cover, which are just sets of 64-bit integers. As an aside, searching for MERRA-2 data is much simpler as there is only one MERRA-2 grid, whereas each MODIS granule features arrays (for longitudes and latitudes) of irregular geolocations. For MERRA-2, once the native array indices are found for a pod, these indices are good for all MERRA-2 data products. STARE enables the harmonization and analysis of diversely arrayed data.

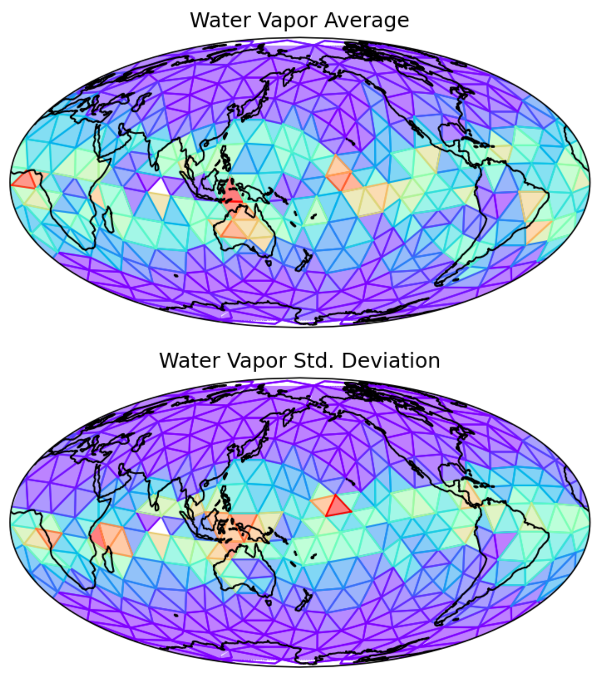

Figure 23. Prototype aggregation of MODIS Level 2 (un-gridded) swath water vapor data associated with granules visualized in Figure 21. Data in legacy file formats were stored in Cloud and processed using the Dask parallel platform. The processing was partitioned using STARE with the loading and calculation performed in parallel. The white patches at the poles are artifacts of the Matplotlib projection and configuration.

Once the data were harmonized on the QL3 pods, calculating statistics for the data in each pod is straightforward and the results are displayed in Figure 23. Again, we note that the harmonized data in the QL3 pods have not been changed, merely re-organized. If we had wished to continue our analysis, or share this data with colleagues, we could have saved the harmonized QL3 pods to Cloud forming a prototype Parallel Optimized Data Store (PODS) of harmonized, analysis-ready MODIS Level 2 swath data. As more diverse data are repackaged in pods and stored in Cloud or HPC environments as PODS, we will be able to drastically increase our ability to perform integrative analysis with STARE-PODS providing the means to combine dramatically different kinds of data at scale.

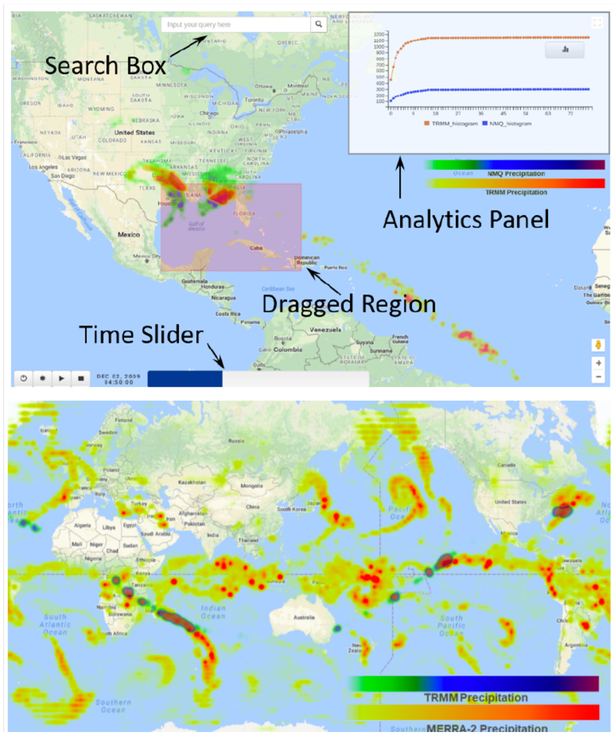

Figure 24. The STARE-enabled web-based graphical user interface for the Optimal Data Layout (ODL) for Scalable Geophysical Analysis in a Data-Intensive Environment enabling interactive exploration of diverse data (MERRA-2, NMQ, and TRMM) stored on a 16-node cluster computer, with responsive big data animated visualization [1].

A final example of how STARE and STARE-PODS can enable new capabilities is the Optimal Data Layout (ODL) Map Interface for the interactive analysis of diverse data sets (Figure 24). In this work, data from 3 Big Earth Science data sets for the winter of 2009-2010 were stored in a SciDB distributed parallel array database management system on a 16-node cluster computer. Harmonizing the data from two different grid layouts and the complex swath data from NASA Tropical Rainfall Measuring Mission (TRMM) spacecraft was almost trivial as the STARE spatial indices fit precisely the sparse array indexing functions of the array database management system. With the STARE-enabled ODL, multiple researchers could simultaneously study (independently or collaboratively) blizzard phenomenon events through responsive, interactive visualizations of the harmonized diverse data including integrated data animations and statistics.

Summary

Today's complex technical systems are interconnected and global, generating vast amounts of geo-spatiotemporal data. Most analyses or products formed from these data have been built on data *homogenized* (interpolated) to common grids, meaning that vast domains of opportunity exist for extracting value from harmonized data. Harmonized data, with its best-possible resolution, physical content, and untrammeled relational context, is critical for the next generation of modeling, forecasting, simulation, and analysis, where interrelationships and causal linkages are key. On the other hand, data homogenization removes important context from observations, fundamentally limiting its usefulness, even as it brings diverse data onto common reference grids.

In this note, we have seen NMQ, MODIS, MERRA-2, SSMIS, GOES, and TRMM data—gridded, swath, imagery, ground-based, space-based—all treated within the same unified, integrative, analytic framework, respecting low-level data integrity, and including an example in Cloud using elastic, scalable processing. STARE's unique capability to scale in both data diversity and volume for a relatively modest investment by stakeholders and end-users is critical for fully realizing the value contained in our geo-data stores.

As the spatiotemporal scales of our observations and simulations become finer and the global linkages of Earth systems become more apparent, the need for scaling in both data diversity and volume is clear. STARE provides a unifying platform for bringing diverse data, data products, and pipelines together in a computationally efficient manner. STARE, though it is perhaps an unfamiliar approach to indexing or organizing Earth Science data, is easy to use and readily extensible to incorporate higher abstractions such as moving objects, STARE dramatically allays the burden, cost, and complexity of data wrangling. STARE-PODS enables the integration of the vast diversity of geo-data products and methods on the distributed parallel computing systems required for scaling to problems of global scope. STARE's simplifying principle of imbuing integers with hierarchical spatiotemporal semantics for indexing eliminates the ever-growing "all-to-all" explosion of data interoperability issues and costs. STARE-based data coordination is vital for making full use the diverse, voluminous geo-data in a host of scientific, commercial, and other contexts.

References

1. Yu, L., M.L. Rilee, Y. Pan, F. Zhu, K.-S. Kuo, H. Yu, "Visual analytics with unparalleled variety scaling for big earth data," IEEE BigData, pp. 514-521 , 11-14 December 2017.
2. https://www.businesswire.com/news/home/20210714005463/en/Global-Geospatial-Analytics-Market-2021-to-2026---Advancements-in-5G-Communications-Technology-Presents-Opportunities---ResearchAndMarkets.com#:~:text=The%20geospatial%20analytics%20market%20size,12.6%25%20during%20the%20forecast%20period.
3. DAOS: Revolutionizing High-Performance Storage with Intel® Optane™ Technology https://www.intel.com/content/www/us/en/high-performance-computing/daos-high-performance-storage-brief.html
4. https://portal.hdfgroup.org/display/HDF5/Virtual+Dataset++-+VDS
5. https://portal.hdfgroup.org/display/HDF5/Single+Writer+Multiple+Reader++-+SWMR
6. HDF5 Virtual File Layer https://support.hdfgroup.org/HDF5/doc/TechNotes/VFL.html
7. Webinar: Learn about New Features in HDF5 1.12.0 https://www.youtube.com/watch?v=cxg-01SMBrI
8. https://bitbucket.hdfgroup.org/projects/HDF5VOL
9. HSDS: Object Store-Based Data Service for Earth System Science https://earthdata.nasa.gov/esds/competitive-programs/access/hsds
10. https://www.hdfgroup.org/solutions/hdf-kita
11. Rilee, M. L., K.-S. Kuo, J. Frew, J. Gallagher, N. Griessbaum, K. Neumiller, and R. E. Wolfe, 2021: STARE into the future of GeoData integrative analysis. *Earth Sci Inform*, https://doi.org/10/gk2d7n.
12. Kuo, K.-S., Y. Pan, F. Zhu, J. Wang, ML Rilee, H. Yu, "A Big Earth Data Platform Exploiting Transparent Multimodal Parallelization," IGARSS 2018 – 2018 IEEE Int'l. Geosci. and Remote Sens. Symp., Valencia, Spain. DOI:10.1109/IGARSS.2018.8518304
13. K.-S. Kuo and M. L. Rilee, "Limiting Data Friction by Reducing Data Download Using Spatiotemporally Aligned Data Organization Through STARE," presented at the 2017 AGU Fall Meeting, 2017.
14. K.-S. Kuo and M.L. Rilee, "STARE Toward Unprecedented Geo-Data Interoperability," in Proc. of the 2017 conference on Big Data from Space (BiDS'17), pp. 90-93, 28-30 November 2017. (Full text available at: https://www.researchgate.net/publication/320908197_STARE_-_TOWARD_UNPRECEDENTED_GEO-DATA_INTEROPERABILITY)
15. J. Gray, A. S. Szalay, A. R. Thakar, G. Fekete, W. O'Mullane, M. A. Nieto-Santisteban, G. Heber, and A. H. Rots, "There Goes the Neighborhood: Relational Algebra for Spatial Data

Search," Microsoft Research Technical Report, MSR-TR-2004-32, April 2004, (arXiv.org. pp. arXiv:cs–0408031, Aug-2004).

16. A.S. Szalay, J. Gray, G. Fekete, P.Z. Kunszt, P. Kukol, and A. Thakar, "Indexing the Sphere with the Hierarchical Triangular Mesh," Micr. Res. Tech. Rpt., MSR-TR-2005- 123, 2005.
17. https://storm.pps.eosdis.nasa.gov/storm/
18. https://arthurhou.pps.eosdis.nasa.gov/Documents/GPM_L1C_DataProductSummaries_V05_V06.pdf