Cloud-Native Architectures for Large-Scale Remote Sensing and Geospatial Data Platforms

Devinder Tokas

Independent Researcher, USA

Abstract

Modern remote sensing platforms use cloud-native architectures with standards-based separation of control and data planes to achieve cost and performance targets at the petabyte scale. Cloud-native stacks are designed around Cloud Optimized GeoTIFF (COG), a byte-range streamable format, SpatioTemporal Asset Catalogs for discoverable metadata management, and Open Geospatial Consortium (OGC) standards for interoperable service delivery. Performance is illustrated in distributed compute engines, including large-scale spatial joins and analyses over multi-temporal data. Service layer design considerations include tile-first APIs and tiled data pipelines, alongside aggressive tile caching at the content delivery network (CDN) and edge layers. Performance considerations model tile latency, time to first pixel, and egress efficiency in distributed systems. These standards enable elastic scalability of platforms and interactive visualization workflows to meet the variability in analytics consumption patterns. Demonstrations have established that performance can be improved with internal tiling, multi-resolution overviews, and HTTP range requests that reduce bandwidth and latency in a wide variety of client ecosystems.

Keywords: Cloud Optimized Geotiff, Spatiotemporal Asset Catalogs, Distributed Geospatial Computing, Tile-First Architecture, Petabyte-Scale Remote Sensing

1. Introduction

Evolving from a single-file repository to a distributed data product, modern remote sensing systems have divergent use cases: serving archives of large-volume rasters, publishing consistent metadata, and generating maps for interactive users under latency restrictions. Much of the work in this area focuses on serving petabyte-scale image collections for analytics and end-user consumption under predictable performance and cost. This requires moving from monolithic map servers to distributed compute engines, from full-file downloads to byte-range streaming, and from proprietary formats to open standards to support many different clients and ecosystems.

The key innovation of the COG format is that it stores raster data in a way that is compatible with HTTP range requests. The OGC Cloud Optimized GeoTIFF Standard version 1.0 was adopted by the Open Geospatial Consortium as an official standard in July 2023 and published as an OGC Standard in October 2023 [1]. The COG specification optimizes the GeoTIFF data structure to provide tiling and multi-resolution overviews that can be efficiently accessed by cloud object storage systems using byte range requests. The COG specification relies on tiling of pixel arrays in a GeoTIFF rather than strips of rows in a file, as well as HTTP GET range requests that allow for delivering bytes based on a requested range of offsets [2].

Using tiling and multi-resolution overviews, a well-formed COG file allows clients to fetch only the appropriate tiles at the appropriate resolution for display in the current viewport. This reduces transfer times and time to first pixel [2]. Well-formed COGs are used for interactive browsing and analysis. For example, the GDAL COG driver for generating overviews is configured to not create overviews if the minimum resolution for which a generated overview is used is greater than 512 pixels [2].

These techniques also anticipate metadata standards such as the SpatioTemporal Asset Catalog (STAC), which provides a standard Item, Catalog, and Collection structure for representing metadata, coupled with standard link relations for traversable search and discovery of space, time, and domain [1]. With the metadata search capabilities of STAC, which standardizes spatial, temporal, provider, licensing, and asset roles as the control plane of remote sensing systems, both clients and compute systems are guided to the authoritative byte in object storage on demand [2]. The service layer is defined by the OGC API standards using modular REST endpoints for tiles and vector features. OGC API - Tiles standardizes access to various tile types (map, imagery, vector, coverage) through tile matrix sets and tile URL templates. It supports predictable addressing schemes for tiles and is designed to have high cache hit rates at edges [2].

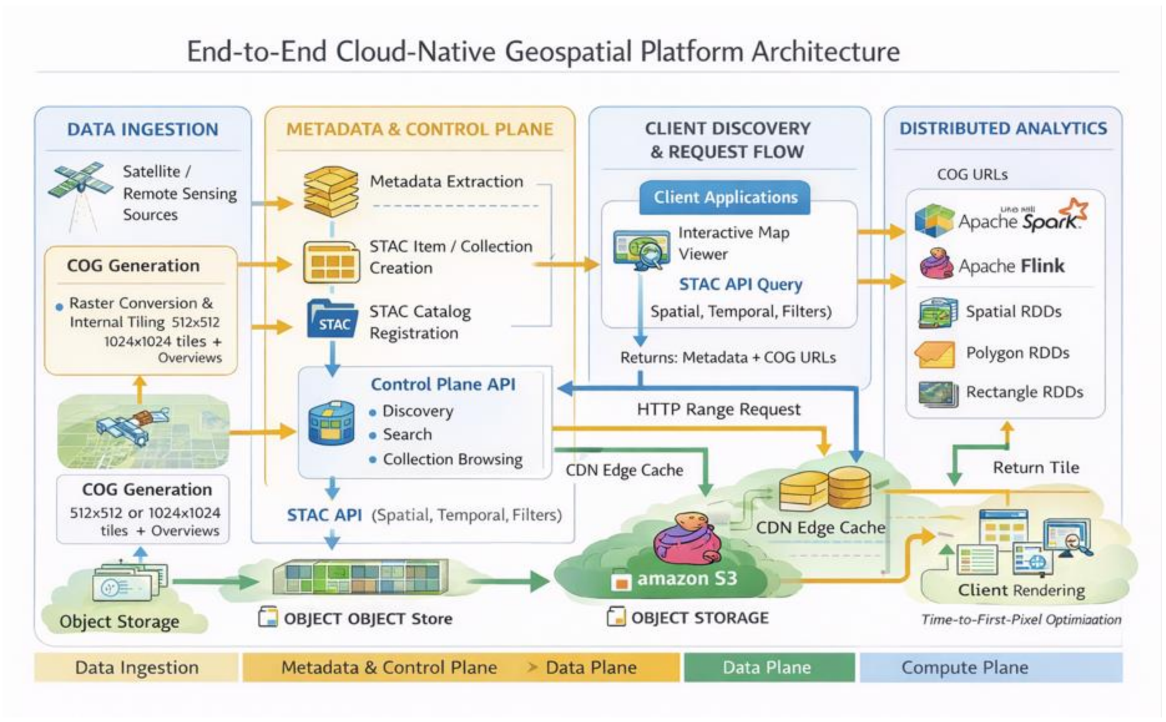


Figure 1: End-to-End Cloud-Native Geospatial Platform Architecture

2. Standards-Based Control and Data Planes

2.1 Architectural Separation

The architecture separates a control and data plane following the cloud-native geospatial interoperability experiments and recommended practices. The control plane manages STAC catalogs, collection metadata, and API definitions, but never serves large raster payloads directly via HTTP(S). It provides governed pointers to the data in object storage, along with metadata to interpret it. This separation allows the control plane to remain responsive and performant even as datasets grow to petabytes.

ARCHITECTURE DIAGRAM 1: Complete Control and Data Plane Separation

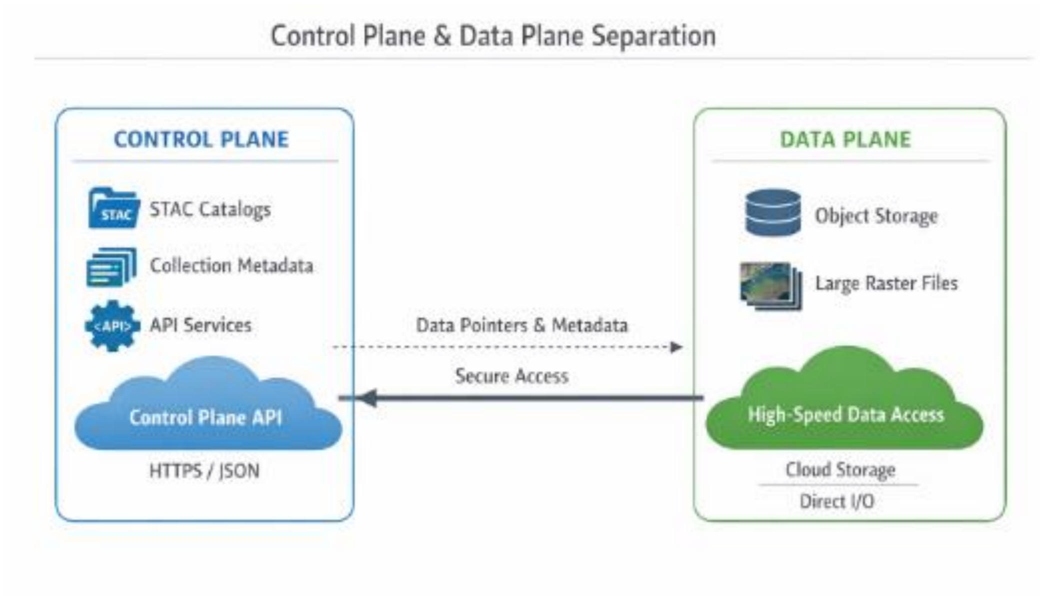


Figure 2: Control Plane & Data Plane Separation

2.2 Validation Through Industry Collaboration

The draft GeoVolumes API specification and implementation were trialed and publicly demonstrated in the OGC Interoperable Simulation and Gaming Sprint, September 21-25, 2020. This demonstrated that control plane implementations can expose discovery and other endpoints implementing the STAC API specifications and standard interfaces for accessing tiles and features. Nine organizations successfully employed server-client implementations of the GeoVolumes API involving multiple data layers, namely CAE, Cesium, Cognitics, Ecere, Helyx, Hexagon, InfoDao, SimBlocks, and Steinbeis. The performance of the control plane architecture was verified as also supporting hierarchical collections and collections of collections.

The San Diego CDB dataset (26 gigabytes) was tested in the southwest corner of geocell N33 W118. The test included 1.7 gigabytes of elevation data provided in the TIF format; 17.2 gigabytes of imagery in the JPEG2000 format; and 6.0 gigabytes of 3D models in the OpenFlight format. The New York City dataset was tested to show that the same discovery mechanisms could be used for the searched metadata, while the actual raster byte streams were in the data plane [3].

Data Type	**Format**	**Size (GB)**
Elevation	TIF	1.7
Imagery	JPEG2000	17.2
3D Models	OpenFlight	6.0
Total		**26.0**

Table 1: San Diego CDB Dataset Composition and Storage Requirements (GB) [3]

The data plane consists of object storage for the COG assets, clustered batch processing, and optimized serving implementations for read-heavy traffic. The model serves paths in eight different server implementations, which were run during the Sprint week. Requests are modeled as traveling from a client to a CDN edge cache, from the edge cache to the tile gateway, which invites the object storage to read the COG windows using range requests. Technology Integration Experiments showed that requests for the tiles were able to achieve interactive frame rates of 40-60 frames per second when the model is batched by the client, although the top-level tiles used to create these images are between 50-100 megabytes in size, which adds one to two seconds of latency before the model is rendered [3].

2.3 Standards Ecosystem Integration

The SpatioTemporal Asset Catalogs specification defines a shared vocabulary using extensible core GeoJSON objects to describe geospatial assets in such a way that catalogs can be compared, and tools can be applied uniformly across multiple catalogs. This is suitable for data naturally associated with space and time, such as satellite imagery [4]. STAC has become the de facto cloud catalog standard. The NASA Earthdata search already uses this lowest common denominator JSON wrapper around Earth data. STAC stays very close to the Cloud Optimized GeoTIFF format [4]. A COG-aware analytic client does not use the tile gateways. The actual conversion times from participants showed approximately 35 minutes for preprocessing to convert the San Diego CDB dataset, and for on-the-fly tile generation, it was a 5-minute startup indexing time, which indexed the bounds of the 6-gigabyte mesh data into octree structures [3].

Standard/API	**Type**	**Primary Function**
GeoVolumes API	Draft specification	Discovery and endpoint exposure
STAC API	Specification	Standard interfaces for tiles and features
STAC	Catalog specification	Shared vocabulary for geospatial assets using GeoJSON
COG	Format standard	Cloud-optimized raster data storage

Table 2: API and Specification Standards [3,4]

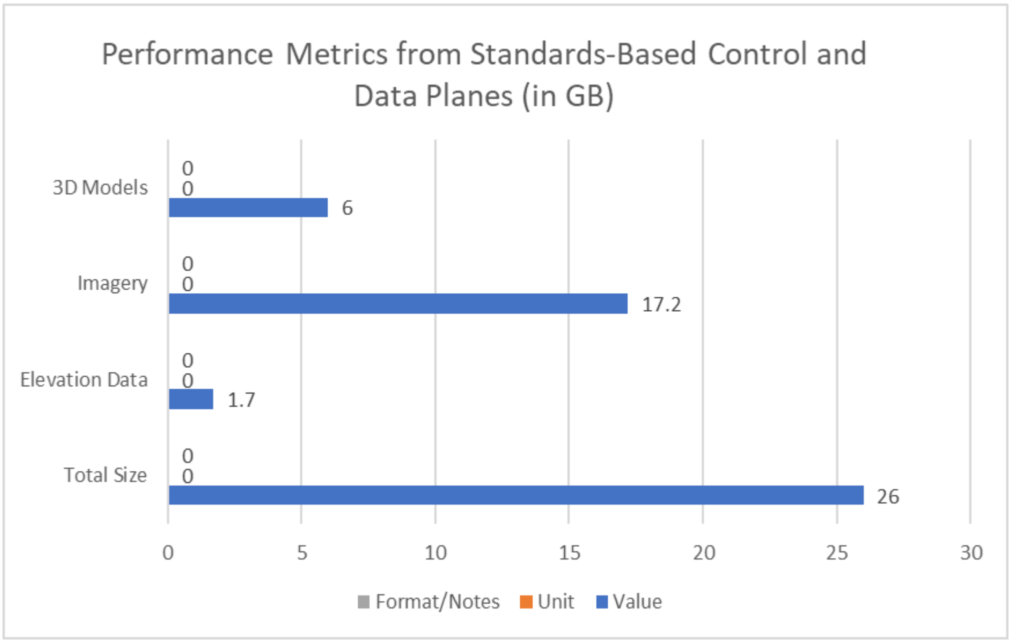


Table 1: San Diego CDB Dataset Composition and Storage Requirements (GB) [3]

3. Distributed Analytics and Vector Processing

3.1 Distributed Compute Architecture

These pipelines often leverage distributed compute engines that have been optimized for geospatial workloads and evaluated on a number of data distributions, including Apache Sedona as an extension of Apache Spark and Apache Flink to support spatial data and spatial SQL. Spatial RDDs join large-scale images' footprints to the area of interest by storing Point RDDs, Polygon RDDs, and Rectangle RDDs in a compact in-memory encoding. Spatial RDDs support ESRI Shapefile, GeoJSON, well-known text, well-known binary, and NetCDF input/output [6].

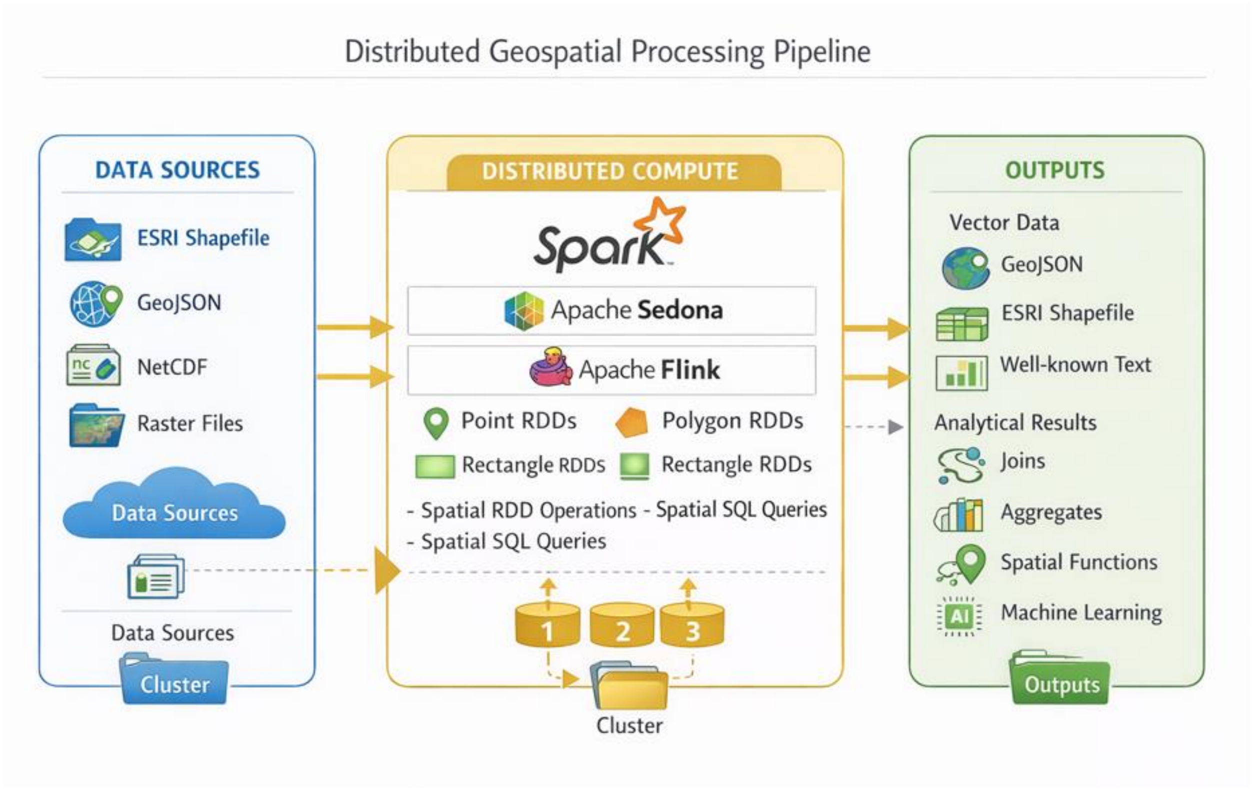


Figure 3: Distributed Geospatial Processing Pipeline

3.2 Spatial Partitioning and Query Optimization

The default join strategies are materializing intermediate tables with time partitions and spatial indexes, and efficiently executing distributed joins of multi-temporal datasets using global spatial data partitioning. The GeoSpark spatial data partitioning algorithm randomly samples the RDD and builds several spatial indexes (KD-Tree, R-Tree, Quad-Tree, etc.) on the sampled RDD. Then, it can quickly repartition the original RDD into spatially close groups of points using the leaf nodes of the spatial index. Each partition of the spatial RDD is distributed, and the local spatial indexes are built for it.

Performance evaluations of GeoSpark show that on a distributed system with Amazon EMR clusters of up to 6 instances, GeoSpark can handle 1 million to 1 billion point records with sizes from 2.39 gigabytes to 38.5 gigabytes of raw data. The time for those evaluations differs based on the underlying storage architecture. Performance summaries for 10 million records under the real-world Natural Earth populated places dataset distribution (7,343 points in 5 MB) include k-NN query, circle query (10 kilometers to 1000 kilometers radius), bounding box query (10x10 kilometers to 1000x1000 kilometers bounding box), and distance join query (0.1 kilometers to 10 kilometers threshold) [6]. Processing times vary depending on the type and parameters of the query.

3.3 Real-Time Processing and Storage Performance

For near real-time applications, spatiotemporal indexes over distributed datastores with proven low latency characteristics on multiple workloads are useful. For example, spatiotemporal queries can be used to detect fire hotspots or track vessels on low-latency event streams. At least one implementation (GeoMesa) works especially well when overlaying high-throughput data feeds for automated response or live map rendering. GeoSpark on MongoDB was 2.08 to 7.84 times slower than HDFS and 2.44 to 8.91 times slower than S3, respectively, using 10 million rows with real-world data distribution and a variety of queries. For larger datasets, performance degradation of between 3.51 and 9.40 times, and between 1.23 and 1.96 times, compared to an HDFS implementation, was reported at dataset sizes of 1 million and 1 billion records, respectively [6].

Operational vector storage implementations make use of PostGIS and offer ACID semantics and advanced spatial indexes, while analytics-ready copies of the data are published as GeoParquet to modern data lakehouse systems. OGC adopted the new Cloud Optimized GeoTIFF Standard, which specifies that COG files must tile all Image File Directories (IFDs) instead of using strips, as in section 15 of the TIFF version 6.0 specification. This allows for much faster access to specific two-dimensional bounding boxes: the relevant data is more compacted, resulting in smaller portions of bytes being read than in the strip case [5].

Experimental results show that the sharding policies are important for MongoDB. When compared to a random sharding policy, the performance of the z-order-based sharding policy achieves 2.42, 1.84, 3.97, and 2.07 times faster speeds for k-NN query, bounding box query, circle query, and distance join query, respectively. This reflects that sharding the data along the space-filling curve can help to organize the data in multiple shard servers and keep the spatial locality [6].

4. Service Layer Design and Caching Strategies

4.1 Tile-First Architecture Philosophy

Interactive mapping systems have become tile-first. The COG can thus be seen as a building block that allows data to live in the cloud, opening up possibilities for parallel processing and not duplicating storage and download costs. The COG was developed by Amazon, Planet Labs, MapBox, ESRI, and the USGS as a new format to put the Landsat Archive on AWS. The discussion on the Landsat-pds mailing list focused on the streaming data format, its use for on-the-fly processing, and how to take advantage of the archive within existing frameworks on Amazon Web Services, without duplication and reprocessing.

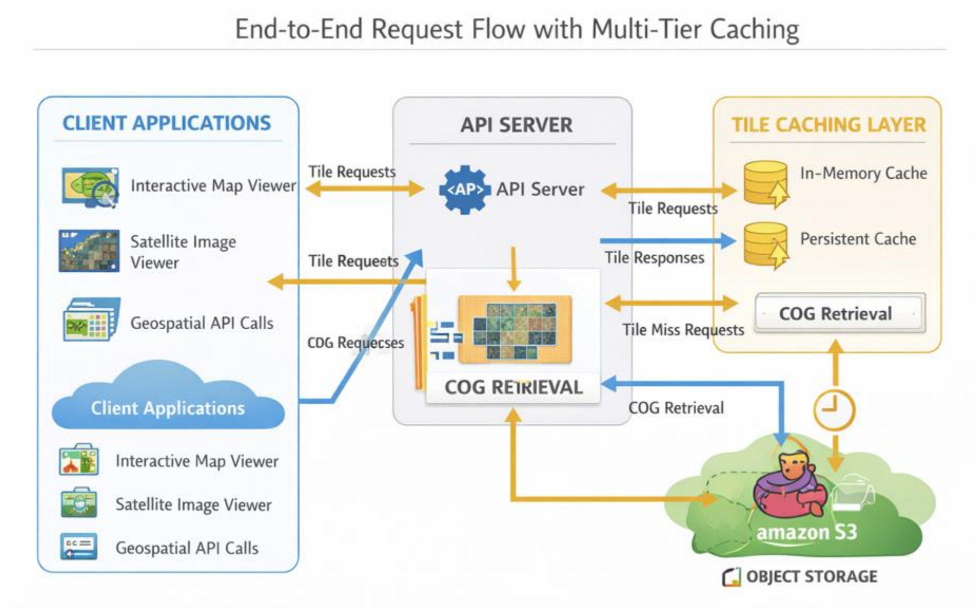


Figure 4: End-to-End Request Flow with Multi-Tier Caching

4.2 Industry Adoption and Implementation Impact

This deterministic pattern-based addressing method to access data in GeoTIFF format through the HTTP Byte Serving protocol allows for aggressive caching at the CDN and edge layer, without hitting the origin for repeat requests. Just as video players skip and rewind by requesting byte ranges without buffering the entire video, COG access works the same way [7]. A hybrid serving execution model was defined, with COGs as the canonical format, tile APIs for general web clients, and governed access to COGs for analyzing workloads.

Organization/Application	Implementation Details
Planet Labs	Converted the entire data pipeline to use COGs
FarmShots	Re-architected domain-specific application as a generalist application using COG format
Santiago & Cintra	Re-architected domain-specific application as a generalist application using the COG format
DigitalGlobe	Integrated COGs into next generation IDAHO system and reprocessed data
OpenAerialMap	Used COGs as part of architectural philosophy for converting user-uploaded data into web-accessible GeoTIFFs

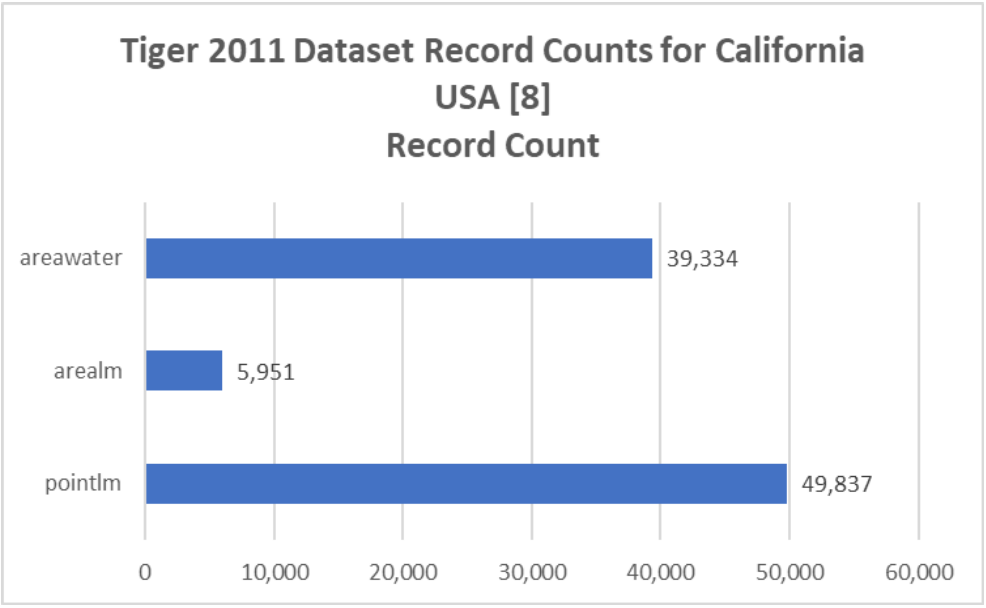
Table 3: COG Format Adoption by Industry Organizations [7]

Planet Labs has converted the entire data pipeline to use COGs. Planet partners FarmShots and Santiago & Cintra have also re-architected their domain-specific applications as generalist applications using the format. Several leading open-source geospatial tools, such as GDAL, QGIS, and GeoServer, have native support for COGs through read support. The latter two require more complex setups [7]. The Tiled Data acquisition pipeline was eventually established as a standard best practice fully supported in GDAL, along with ample documentation and performance benchmarks on the GDAL Wiki. DigitalGlobe integrated COGs into its next-generation IDAHO system and reprocessed their data. Furthermore, OpenAerialMap embraced the concept of COGs as part of their entire architectural philosophy of converting user-uploaded data into web-accessible GeoTIFFs, which were then streamed as tiled imagery to clients—all showing that modern on-the-fly processing can be performed in seconds at the scale of hundreds and thousands of computers rather than days and weeks [7].

4.3 Reliability and Performance Benchmarking

Vector overlays (e.g., scene footprints or derived change polygons) are served via APIs with support for spatial filters and bounding box queries. When vector tiles are used for web-scale rendering, they provide a compact and style-flexible alternative with similar cache hit ratios to raster tiles. Reliability measures include using request coalescing of multiple requests for the same tile, per-tenant quotas, and adaptive degradation using lower resolution overviews during peak utilization to reduce request amplification to the object storage.

Performance evaluations of Big Spatial Data using benchmark workloads of spatial join operations, range queries, and spatial analysis functions reveal that multiple performance results may be observed in distributed systems. The tests were performed on an 8-machine cluster with an Intel Xeon CPU E5472 at 3.00 GHz, a two-socket dual processor with 4x2 cores, 16 gigabytes of RAM, and a 500 gigabyte HDD running Ubuntu 14.04 64-bit OS, Oracle JDK 1.8.0_81, Hadoop-2.3.0, Spark-2.1.1, and Apache Ignite Fabric 1.2.0 [8]. Real spatial datasets from the Tiger 2011 release for California, USA, as pointlm (49,837), arealm (5,951), areawater (39,334), and edges (4,173,498), show that SpatialIgnite's best speedup is 1.92x for the LineIntersectsArea join query and with a minimum speedup of 1.84x for LineWithinArea for scaling from 4 to 8 clusters. Scalability results showed that SpatialHadoop performed better with 4 nodes; however, performance degraded with more nodes due to the disk-based architecture. Range query performance results showed that SpatialIgnite outperformed other systems except for the line datasets, where the performance difference in all compared systems was similar [8].



Graph 2: Tiger 2011 California Dataset Size by Feature Category (Record Count) [8]

5. Performance Metrics and Service Level Objectives

5.1 Comprehensive Performance Framework

The proposed service level objectives provide a thorough framework to measure performance against technical, user experience, and cost governance by systematically measuring distributed computing infrastructures. Tile latency targets are systematically measured across the 50th, 95th, and 99th percentiles per zoom level and region with clusters of eight machines with an Intel Xeon CPU E5472 (3.00 GHz) x2 (dual) 4 (four cores) each, 8 cores per machine, with 16 GB RAM and 500 GB hard disk drives running Ubuntu 14.04 64-bit. This demonstrated execution times of spatial operations varying greatly, depending on available storage architecture and query types.

5.2 Spatial Query Performance Analysis

The database was evaluated by running 14 types of pairwise spatial join queries on a set of spatial datasets, namely pointlm (49,837 point features, such as airports and movie theaters), arealm (5,951 polygon features, such as boundary areas of cities, forests, rivers, and national parks), areawater (39,334 polygon features, such as water features including lakes, rivers, and ponds), and edges (4,173,498 line features, such as infrastructure including roads, rivers, and railways) [9].

Experiments showed that SpatialIgnite consistently performs better than disk and in-memory implementations. SpatialIgnite speedups ranged from 1.84x (LineWithinArea Spatial Join) to 1.92x (LineIntersectsArea Spatial Join) between designs in cluster scalability tests from four-node and eight-node clusters. Range queries were comparable across implementations for line datasets and considerably faster for point and polygon datasets, ranging from seconds for smaller datasets to hundreds of seconds for complex spatial join queries with large datasets [9].

5.3 Time-to-First-Pixel Optimization

For Cloud Optimized GeoTIFF-aware visualization workflows, the time-to-first-pixel metric includes both the latency time of the first response and the number of bytes transferred to download the first visual representation of the image. The COG specification dictates that Image File Directories be tiled and organized into 512x512 or 1024x1024 pixel tiles and include reduced resolution subfiles for which the resolution is reduced by at least a factor of two to a maximum factor of ten from the previous level to enable progressive rendering through the transmission of only the bytes containing the tiles within the visible viewport [10].

COG partial reads are validated to check that the same number of bytes is saved as would have been read with full-file access. HTTP servers serving COGs should support byte range requests and advertise Access-Control-Allow-Headers: range on HTTPS cross-origin responses. This allows efficient downloading of image subsets and grid coverage data subsets over the web to rapidly visualize and efficiently process geospatial data [10].

Conclusion

The scalability of cloud-native remote sensing platforms relies heavily on the separation of concerns in architecture and implementation and standards-based composition, which allows for independent replacement or evolution of components without changing interfaces. Internally tiled Cloud Optimized GeoTIFFs enable streaming access to archived raster imagery using byte-range requests for arbitrary regions, while SpatioTemporal Asset Catalogs offer discoverable metadata for guiding client applications to authoritative storage endpoints for large payloads. OGC API standards define interoperable serving interfaces that create deterministic, addressable tile grids for aggressive caching at the network edge to absorb repeated requests and reduce load on permanent storage. In terms of elastic processing of large-scale spatial analysis tasks, distributed engines enable the separate evolution of storage engines and compute runtimes under a shared set of contractual interfaces through the division of the control and data planes. The demonstrated impact includes achieving interactive frame rates for real-time visualization, dramatically reducing processing times from days-weeks to seconds through massively parallel computing, enabling significant speedup through cluster scaling, and delivering substantial performance improvements through optimized sharding policies across spatial query types. The architecture thus allows for continual advancement in sensor technology, product needs, and user expectations, as well as management of an ever-expanding petabyte-scale archive of Earth observation data and presentation of respective interactive map experiences and analytical capabilities for multiple client ecosystems, transforming geospatial platforms from monolithic systems requiring full-file downloads to distributed, standards-based architectures supporting byte-range streaming with high cache hit rates at network edges.

References

[1] Nathan Pollack, "Cloud Optimized GeoTIFF (COG) File Format," NASA Earth Science Data and Information System Standards Office, 2024. [Online]. Available: https://www.earthdata.nasa.gov/s3fs-public/2024-05/ESDS-RFC-049%20Cloud%20Optimized%20GeoTIFF%20V1.pdf

[2] Joan Maso, "OGC Testbed-17: Cloud Optimized GeoTIFF Specification Engineering Report," Open Geospatial Consortium, 2008. [Online]. Available: https://docs.ogc.org/per/21-025.pdf

[3] Thunyathep Santhanavanich, et al., "Open Geospatial Consortium (OGC) Interoperable Simulation and Gaming Sprint Engineering Report," ResearchGate, 2020. [Online]. Available: https://www.researchgate.net/publication/357394775

[4] D. J. Newman and ESDIS Standards Coordination Office, "SpatioTemporal Asset Catalogs (STAC)," NASA Earth Science Data and Information System Standards Coordination Office, 2023. [Online]. Available: https://www.earthdata.nasa.gov/about/esdis/esco/standards-practices/stac

[5] Open Geospatial Consortium, "OGC Cloud Optimized GeoTIFF Standard," 2023. [Online]. Available: https://docs.ogc.org/is/21-026/21-026.html?utm_source=chatgpt.com

[6] Hansub Shin, et al., "A comparative experimental study of distributed storage engines for big spatial data processing using GeoSpark," PubMed Central, 2021. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC8246422/?utm_source=chatgpt.com

[7] Chris Holmes, "Cloud Native Geospatial Part 2: The Cloud Optimized GeoTIFF," Medium, 2017. [Online]. Available: https://medium.com/planet-stories/cloud-native-geospatial-part-2-the-cloud-optimized-geotiff-6b3f15c696ed

[8] Md Mahbub Alam et al., "A Performance Study of Big Spatial Data Systems," in Proc. 7th ACM SIGSPATIAL Int. Workshop Analytics Big Geospatial Data (BigSpatial 2018), [Online]. Available: https://www.cs.unb.ca/~sray/papers/BigSpatialBenchmark_BigSpatial2018.pdf

[9] Md Mahbub Alam, et al., "A Performance Study of Big Spatial Data Systems," ACM Digital Library, 2018. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/3282834.3282841

[10] Swiss Geoinformation Strategy, "Cloud Optimized Geospatial Formats: Status Report," GeoStandards.ch, Mar. 2024. [Online]. Available: https://download.opengis.ch/Cloud_Optimized_Geospatial_Formats-Report.pdf