Serverless Data Storage for Scalable Geospatial Applications in the Cloud

Holderness T[1*], Varley M[1]

[1]Addresscloud

December 18, 2020

Summary (100 words)

This paper presents a web service that combines recent developments in geospatial indexing with serverless cloud storage to provide a scalable architecture for querying geospatial data. The service uses a key-value database, indexed using the H3 spatial index as a cache for multiple geographical data-sets. In instances where cached data is unavailable, secondary raster and vector services provisioned using Cloud Optimised GeoTiffs and a scaleable PostGIS database can be queried with point or polygon geometries. The service is used to provide Addresscloud's Intelligence API, which is a global service that provides property-scale geographic risk assessments to insurance underwriters.

KEYWORDS: serverless, architecture, geospatial, database, insurance

1. Introduction

Serverless cloud infrastructure enables the creation of geospatial applications that scale in real-time in response to the number of requests received (e.g. Bebortta et al., 2020; Lehto et al., 2020). However, serverless geospatial applications are restricted by the limitations of current spatial databases, and the speed at which they can scale to meet the demands of multiple simultaneous requests. Non-spatial serverless applications frequently use both key-value databases and object stores as scalable storage. Recent advances in geographic data specifications enable both vector and raster data to be stored as objects in the cloud (e.g. Lehto et al., 2020; Netek et al., 2020). For example, the Mapbox Vector Tiles (MVT) and Cloud Optimised GeoTiff (COG) formats provide both spatial indexing functionality through tiling, and cloud-native access to data (e.g. via HTTP requests) suitable for object storage. Similarly, new spatial indexing systems such as H3 (Sahr 2008) enable efficient encoding of geographic location that facilitates storage of spatial data in serverless key-value databases. Concurrently, cloud providers, recognising the value of providing infrastructure for geospatial applications, have developed the first generation of spatial relational databases capable of scaling capacity in near-real time (e.g. AWS Aurora Serverless for Postgresql and Google's Big Query).

Previous studies have examined the utility of these developments independently (Kraft et al., 2019; Bebortta et al., 2020; Lehto et al., 2020; Netek et al., 2020). This paper presents a combined architecture that uses three types of serverless data-store (key-value database, tiled object storage, and scalable spatial database) to build a serverless geospatial web service. The integration of multiple serverless data-stores in this manner to form a GIS capable of querying both raster and vector data has not been previously reported in the literature. This architecture was developed for the Addresscloud Intelligence service which is used to provide property-scale geographic risk assessments to insurance underwriters for a range of hazards including flood, windstorm, fire, wildfire, and subsidence.

[1]* tomas@addresscloud.com

2 Serverless Architecture

Addresscloud is a geocoding and property intelligence service. The Addresscloud Intelligence API provides property and geographic risk information to insurance underwriters operating through the Lloyd's Market in London. The API powers client applications and the Addresscloud Maps tool (Figure 1) - a web-based map application to visualise property and geographic risk data. Users query the service by providing a unique property reference, a coordinate pair, or a polygon. The API then returns attributes from multiple data-sets for the given property or location (Figure 1). Attributes include fire block ratings, crime scores, property type, and geographic perils such as flood, wildfire, and subsidence risk. The Intelligence service was architected to meet three requirements: (a) scale seamlessly in response to the number of requests without impacting performance; (b) transactions for known locations should be performed sub-second; (c) the data-store must be readily expandable for new data.

Figure 2 shows an overview of the service architecture, consisting of a HTTP API, backed by an AWS Lambda function which contains the service’s core logic. The function scales horizontally, and parallel executions are directly attributable to the number of concurrent requests received by the API. This function has access to three types of data-store against which spatial queries are executed to service incoming requests;

(1) a cache in the form of a key-value database, indexed using H3
(2) a vector service backed by a serverless PostGIS database
(3) a raster service backed by Cloud Optimised GeoTiffs stored as objects

The following sections describe each of these data-stores, and their underlying technologies.

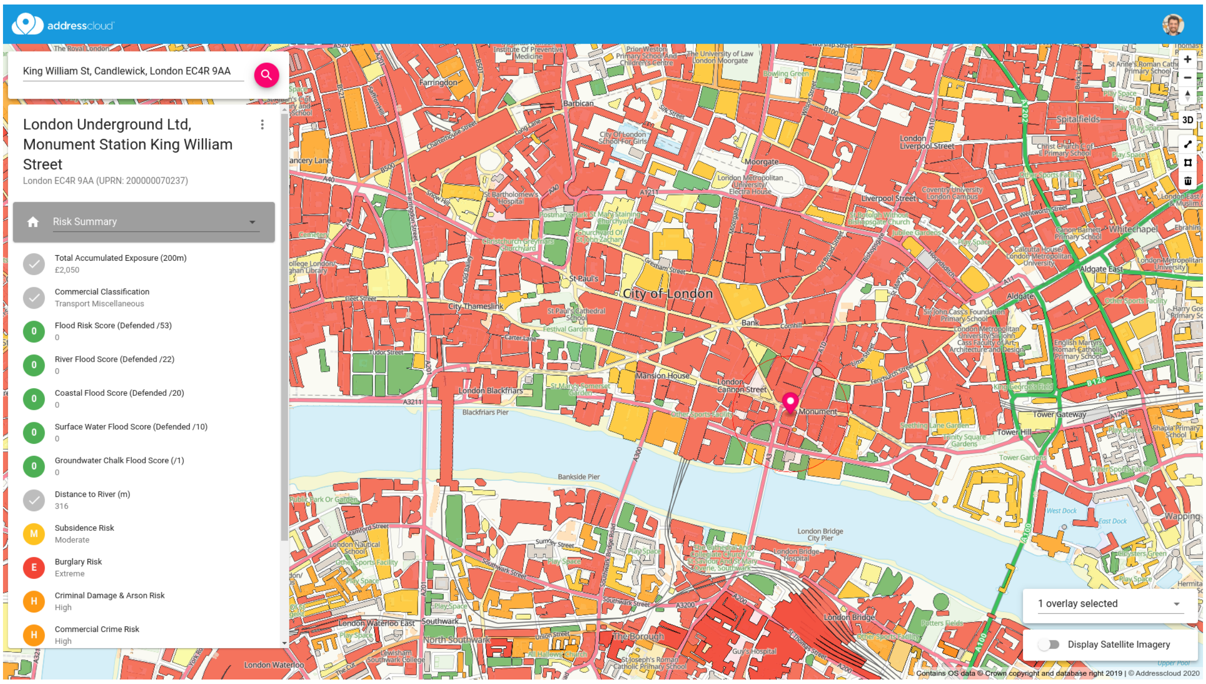

Figure 1 Screenshot of the Addresscloud Maps application showing fire-rating blocks for the City of London. The data in the application is sourced from the Intelligence API.

2.1 H3 Spatial Index for Key-Value Storage

The H3 geospatial index is a hexagon-based tiled representation of the Earth's surface (Sahr et al,. 2003, Sahr 2008). The index is capable of providing a unique hexagon identifier for any location of the Earth's surface at a variety of different scales and has been widely reviewed in the literature (Sahr 2008; Bondaruk et al., 2019; Kraft et al., 2019; Li et al., 2020). H3 is used to index the Addresscloud Intelligence service cache. The cache works by pre-processing data-sets of known locations and their attributes. The H3 index is used as a common identifier across all data. A serverless key-value datastore is populated with attributes keyed by H3 identifier. When an API request is received that contains a known location the service converts the location using H3, and queries the cache (see Figure 2). The key advantage of H3 is its ability to consistently provide a global index of identifiers for multiple spatial layers, and to determine the H3 identifier for a given location with millisecond latency.

2.2 Spatially Indexed Raster Data

In the event that the cache cannot be queried (for example when a user provides arbitrary polygon) the service must query data-sets from the raster and/or vector services (see Figure 2). When compared to relational databases, object storage provides a cloud-native, inexpensive, robust and scalable medium for spatial data storage (Holderness, 2019). The Cloud Optimised Tiff specification (COG) is a standard for internal organisation of a GeoTiff file to support consumption of the data via a HTTP connection (Cloud Optimized GeoTIFF, 2020). The raster service (Figure 2) uses the Rasterio library to convert real-world geometries to pixel coordinates and requests the relevant pixel values from a COG stored in an AWS S3 bucket. In this configuration, COGs stored in an S3 bucket provide a scale-free data-store for raster data (up to the AWS S3 imposed limit of 5,500 requests per second). Once the Raster service successfully queries the required pixel values it returns to the Intelligence function, whereupon they can be used to complete the API request.

2.3 Serverless Spatial Relational Database

To provision queries of vector data a PostgreSQL database is used. The database is hosted as an AWS Aurora Serverless database which is architected so that data storage and database compute capacity are separated. The advantage of this architecture is that the compute capacity can be scaled up and down in response to load placed upon the database system by the number and complexity of queries (Abraham, 2016). Aurora serverless for PostgreSQL was chosen as it enables the use of the widely used PostGIS extension, which provides extremely efficient indexing and manipulation of spatial data via SQL queries. Additionally, Aurora Serverless provides a data API for the automated management of connection pools which cannot be easily incorporated into stateless functions, helping reduce query latency. Similar to the Raster service, the Vector service constructs queries from the geometry provided and executes them against relevant data-sets in the database.

3 Conclusions and Future Research

The Addresscloud Intelligence service uses a combination of serverless data-stores to provide a scalable API for geospatial data. In comparison to Addresscloud's previous non-serverless spatial data-stores (e.g. PostGIS/PostgreSQL running on virtual servers) the new architecture has demonstrated consistent performance (99.9% of queries resolved within 500ms) which is decoupled from the number of concurrent queries. Currently, the biggest restriction is the time it takes the vector service to scale under high-load. In December 2020 AWS announced Aurora Serverless v2 which promises to provide millisecond scaling. Future research is needed to evaluate the performance of Aurora Serverless v2 when used for spatial queries.

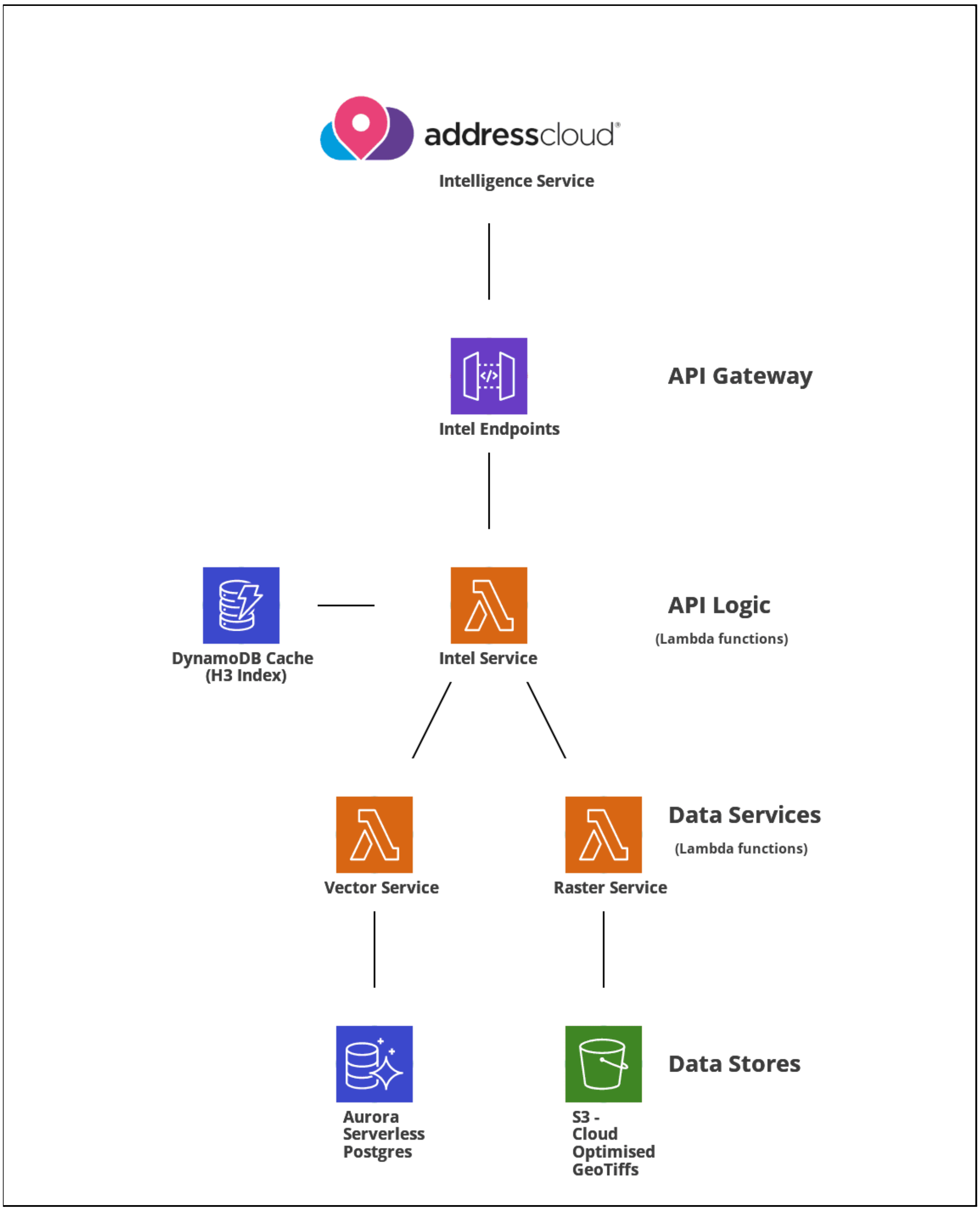

Figure 2 Architecture of the Addresscloud Intelligence Service.

References

Abraham, S., 2016. *Introducing the Aurora Storage Engine | Amazon Web Services*. [online] Amazon Web Services. Available at: https://aws.amazon.com/blogs/database/introducing-the-aurora-storage-engine [Accessed 15 December 2020].

Bebortta, S., Das, S.K., Kandpal, M., Barik, R.K. and Dubey, H., 2020. Geospatial serverless computing: Architectures, tools and future directions. *ISPRS International Journal of Geo-Information*, *9*(5), p.311.

Bondaruk, B., S. A. Roberts, and C. Robertson. "Discrete Global Grid Systems: Operational Capability of the Current State of the Art." In *Proceedings of the 7th Conference on Spatial Knowledge and Information Canada (SKI2019)*, vol. 2323. 2019.

Cogeo.org. 2020. *Cloud Optimized GeoTIFF*. [online] Available at: https://www.cogeo.org [Accessed 14 December 2020].

Holderness, T., 2019. *A COG In The Machine - Using Cloud Optimised GeoTiffs to Query 24 Billion Pixels In Real-Time*. Presentation at Free and Open Source Software for Geospatial (FOSS4G) conference, Bucharest.

Kraft, R., Birk, F., Reichert, M., Deshpande, A., Schlee, W., Langguth, B., Baumeister, H., Probst, T., Spiliopoulou, M. and Pryss, R., 2019, June. Design and implementation of a scalable crowdsensing platform for geospatial data of tinnitus patients. In *2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)* (pp. 294-299). IEEE.

Lehto, L., Kähkönen, J., Oksanen, J. and Sarjakoski, T., 2019. Flexible Access to a Harmonised Multi-resolution Raster Geodata Storage in the Cloud.

Li, M. and Stefanakis, E., 2020. Geospatial Operations of Discrete Global Grid Systems—a Comparison with Traditional GIS. *Journal of Geovisualization and Spatial Analysis*, *4*(2), pp.1-21.

Netek, Rostislav, Jan Masopust, Frantisek Pavlicek, and Vilem Pechanec. "Performance Testing on Vector vs. Raster Map Tiles—Comparative Study on Load Metrics." *ISPRS International Journal of Geo-Information* 9, no. 2 (2020): 101.

Picoli, M.C., Simoes, R., Chaves, M., Santos, L.A., Sanchez, A., Soares, A., Sanches, I.D., Ferreira, K.R. and Queiroz, G.R., 2020. Cbers Data Cube: a Powerful Technology for Mapping and Monitoring Brazilian Biomes. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *3*, pp.533-539.

Sahr, K., White, D. and Kimerling, A.J., 2003. Geodesic discrete global grid systems. *Cartography and Geographic Information Science*, *30*(2), pp.121-134.

Sahr, Kevin. "Location coding on icosahedral aperture 3 hexagon discrete global grids." Computers, Environment and Urban Systems 32, no. 3 (2008): 174-187.

Biographies

Tomas is the CTO of Addresscloud where he leads research and development to create risk mapping services. Prior to joining Addresscloud Tomas led the Urban Risk Map project at the MIT Urban Risk Lab. Tomas has a PhD in Geoinformatics from Newcastle University and is a Chartered Geographer (CGeog).

Mark Varley is Addresscloud's CEO and Founder, and has extensive experience building geospatial applications for the financial and insurance sectors. Mark has previously worked for Accenture, Royal Bank of Scotland and RSA Insurance.