GSKY: A scalable, distributed geospatial data-server

Pablo R. Larraondo, Sean Pringle, Joseph Antony, Ben Evans
pablo.larraondo@anu.edu.au

National Computational Infrastructure, Australian National University
ACT 2601, Australia

Abstract

The rapid growth of earth systems, environmental and geophysical datasets poses a challenge to both end-users and infrastructure providers. GSKY is a scalable, distributed server which presents a new approach for geospatial data discovery and delivery using OGC standards. In this paper we discuss the architecture and motivating use-cases that drove GSKY's design, development and production deployment. We show our approach offers the community valuable exploratory analysis capabilities, for dealing with petabyte-scale geospatial data collections.

1 Introduction

Earth systems, environmental and geophysical datasets are extremely valuable sources of information capturing both the state and evolution of the Earth. Many disciplines and applications require initial input data to be pre-processed in multiple ways before it can be used (Schnase et al., 2016).

For researchers experimenting with algorithms across large datasets or combining multiple data sets, the traditional approach is often to batch-process data and store the output for subsequent analysis. This approach is now rapidly becoming infeasible, due to the amount of storage space and data transformation work that it requires.

Recent developments in distributed computing using interactive access to significant cloud infrastructure opens the door for new methods of processing data on-demand and in near-realtime. We make the case that this approach alleviates the need to store individual copies of each derivative product i.e trading compute for storage, when feasible.

The National Computational Infrastructure (NCI), hosted at the Australian National University (ANU), has over 10 Petabytes of nationally significant research data collections (Evans et al., 2015). These collections, covering a range of environmental data, imagery and modelling output, are now made available via a scalable, distributed geospatial data-server, called GSKY (pronounced [jee-skee]) using OGC standards-compliant interfaces.

GSKY is designed from the ground-up to support on-demand processing of large geospatial data products such as satellite earth observation data, as well as numerical simulation output, thus allowing interactive data exploration using flexible user-defined functions. It dynamically and efficiently distributes the requisite computations amongst cloud nodes, thus providing a scalable analysis framework. In many cases this approach obviates the need to pre-process and store derivative products.

Typical data wrangling problems such as managing different file formats and data types, or harmonising coordinate projections and spatio-temporal resolutions, are handled transparently by GSKY. It is agnostic of

any underlying file format or data organisation. This is achieved by decoupling the data ingestion and indexing process as an independent service. An ingestion service crawls collections either locally or remotely by extracting, storing and indexing all spatio-temporal metadata associated with each individual record or data-collection.

GSKY has functionality for specifying how ingested data should be aggregated, transformed and presented. It presents an OGC standards-compliant interface, allowing readily accessible data for users via Web Map Services (WMS), Web Processing Services (WPS) or underlying source data via Web Coverage Services (WCS). We will also present use-cases where we have used these new capabilities to provide a significant improvement over previous approaches.

This paper is structured as follows: Section 2 provides background and reviews previous work. Section 3 describes the motivating use-cases that drove the development of GSKY. Section 4 gives a high level architecture view into GSKY. The paper concludes and proposes future developments.

2 Background and Previous Work

Amongst all available geospatial data server implementations, THREDDS (Caron and Davis, 2006) and GeoServer (Deoliveira, 2008) have been popular choices for exposing and interacting with data in the climate and earth observation communities.

THREDDS provides remote access to geospatial data stored in scientific multidimensional file formats such as NetCDF, GRIB or HDF. It implements a binary serialization protocol, OPeNDAP (Cornillon et al., 2003), which allows users to efficiently request the contents of files or subsets of them. It also provides capabilities to aggregate several files along the spatial and temporal dimensions into a single entity using a markup language called NcML. However, these capabilities are limited and currently do not scale well when the number of files grows or when dealing with sparse data.

GeoServer has been, for many years, the tool of choice to serve geospatial data to the earth observation and more generally GIS communities. It uses OGC standards to present data to the user in either rendered images (WMS), gridded numerical values (WCS), geometries (WFS) and also offers the possibility to perform analysis operations (WPS). GeoServer has the ability to abstract the notion of map projections, allowing users to consume the same data in different projections where the requisite transformations occur in real time, server-side.

Currently GeoServer is unable to provide services directly from source files that form a data-collection. It is dependent on pre-generation of an internal representation of the data at different resolutions which are known as pyramids[1]. These pyramids accelerate access to the underlying data-collection, at the expense of having to compute and store the pyramids beforehand. When the size of the collection being served grows, the effort to generate and store these internal products also increases, often leading to significant management overheads.

We note that the data-management models implemented by THREDDS and GeoServer are currently limited by a) the lack of features which allow easy and performant methods to create aggregations over collections of files (as in the case of THREDDS), or b) the generation of views over data in real time from the original files without requiring any intermediate product generation (as in the case of GeoServer).

Both solutions have been designed to operate as a single server. Hence, the method of scaling up their respective operational response capacity is often by improving the hardware capabilities of the underlying hosting machine. This causes cost constraints, eventually limits data that can be served and impedes the end-user experience.

As the size of geospatial data collections grows, there is a corresponding need for solutions which allow for the serving of information directly from underlying 'raw' files. Hand-in-hand with this, one needs to scale-out services by using clusters of machines to perform the required aggregations and transformations both on-demand and in a robust, fault-tolerant manner. Recent projects have appeared to address this problem and we review some of these key efforts.

The EarthServer project[2] is an international project funded through the European Union (EU) encompassing multiple organisations from different countries. The Array Database System, RasDaMan (Baumann et al., 1998), a raster database management system, is the underlying technology used by the project. RasDaMan can serve large collections of satellite and climate data using OGC standards. It offers comprehensive ingestion mechanisms allowing users to serve different kinds of data, as well as describing and performing complex server-side analytics using the WCPS[3] standard. However, the open source version of RasDaMan cannot use external 'raw' data files

[1] http://docs.geoserver.org/stable/en/user/tutorials/imagepyramid/imagepyramid.html
[2] http://www.earthserver.eu/
[3] http://www.opengeospatial.org/standards/wcps

as its source data - it requires data to be fully ingested into its backing database. This limitation is solved in the commercial version, however there is not much published information available on the formats and conventions that the files require and how it implements its distributed computing methods.

The Australian Geoscience Data Cube (AGDC) is another project working on the goal of achieving a general geospatial data platform (Lewis et al., 2016). AGDC is presented as an open-source platform enabling public access to data from several earth observation satellites, and offers analytics capabilities, for example, to study temporal trends over user defined geographic areas.

GeoTrellis (Kini and Emanuele, 2014), is a relatively new open-source project for enabling geospatial processing based on the MapReduce architecture. This project builds on top of the available algorithms in Apache Spark, allowing for batch and interactive processing on large geospatial datasets.

Google Earth Engine (GEE) (Gorelick, 2013) can be considered the model of a production ready, large scale, distributed geospatial data server. Scientists and users from the general public are relying on it for research outputs, as well as to rapidly visualise or analyse temporal trends. GEE offers a complete and well defined graphical user interface and programmatic access via its Python and Javascript APIs. Users can define their analysis workflow using these interfaces and submit it to the backend infrastructure. The required computations are distributed among Google's cloud infrastructure and results are gathered and returned to the user within a time determined by the complexity of the operation and the billing policy of the user. Apart from the public APIs, there is little information available about the back-end infrastructure and distribution model used by the GEE.

3 GSKY's Motivating Use-Cases

The motivation behind the development of GSKY has been driven by the need to expose curated satellite imagery collections and climate simulation output to NCI's user community. The NCI has experienced significant demand from researchers wanting to combine data from different sources, as well as rapidly test new computational models/algorithms. Often, data fusion processes require inputs to be normalised to allow pixel to pixel comparisons between different datasets. This process imposes significant challenges without knowing *a priori* the different conventions, storage/file formats, map projections and data-types as used in the curated data collections hosted at the NCI.

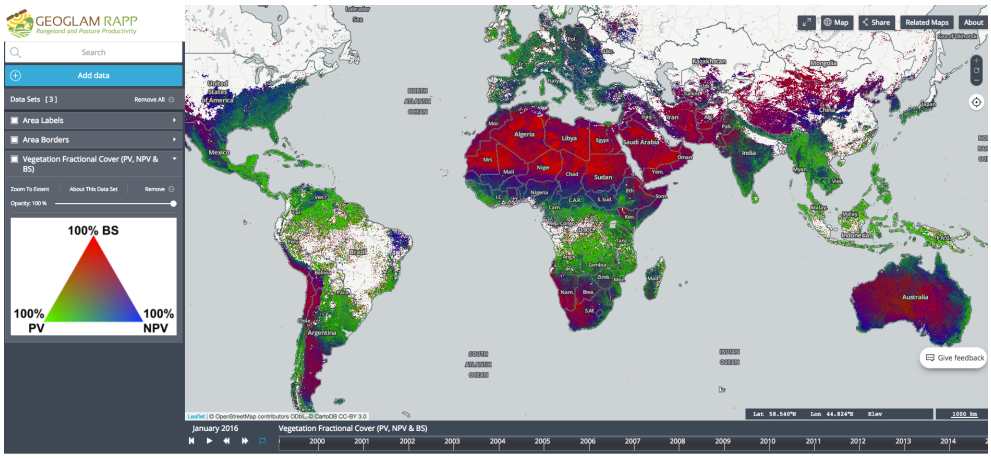

Figure 1: Fractional Cover data being computed and served by GSKY from original MODIS sinusoidal tiles for GEOGLAM RAPP

Another major factor for a unifying methodology to work with geospatial data collections is from the machine learning community. Recent advances in image recognition based on deep learning techniques have opened an exciting path to explore new ways of using geospatial data (Marmanis et al., 2016). Specific hardware and software to run these techniques is becoming available, but the underlying machine learning algorithms require access to large collections of ready-to-consume data, which at present is a limiting factor to uptake. GSKY aims to reduce this gap by offering geospatial data-as-a-service which can be consumed by these emerging applications.

A key project which helped mature GSKY's production deployment is the GEOGLAM Rangeland and Pasture

Productivity (RAPP) project (Guerschman et al., 2015)[4]. The project is a joint initiative between CSIRO and the Group on Earth Observations (GEO), aimed at monitoring the condition of the worlds rangelands and pasture lands in real-time. This project requires synthesis of several satellite products based on MODIS as well as modeled climatic data for visualization and interactive analysis. The project uses the National Map front-end infrastructure (Belgun et al., 2015) to expose the data and connect to the services that GSKY provides at the back-end. Figure 1 is a screen-shot of the production GEOGLAM RAPP portal containing a representation of the different kinds of vegetation covering the planet. Using this application, end-users can interactively visualise and perform regional time series analysis for different satellite products. Apart from MODIS satellite data, GSKY has been validated with the following collections: ERA-Interim, CHIRPS-2.0, Himawari 8 and global Landsat 8.

4 GSKY's Architecture

GSKY posits a model of generating meaningful end-user products from the underlying original source data, without the need to store intermediate products or pre-compute results. Geospatial analysis often involves a series of data extraction and image transform operations, which can be modeled as a transformation process of the original data into a specific product. These operations can be abstractly represented as a sequence of independent processes in which data flows along a pipeline structure producing the final result in the end, i.e a Directed-Acyclic-Graph (DAG). DAGs capture workflows, which in themselves are defined as a composition of individual processes which are connected together forming nodes in the graph and which ultimately result in an output product.

GSKY implements the following OGC standards: WMS, WCS and WPS. These protocols provide a well defined interface for the input requests as well as the output responses. Parameters specified in these protocols are used as inputs to feed the previously described workflows whose outputs are then returned to the user, using standards-prescribed output formats. The implementation of these workflows are generic, hence allowing GSKY to be easily extended to support Open Street Maps or OPeNDAP (Cornillon et al., 2003) in the future.

The following sections focus on two fundamental components of GSKY. The first section, the indexing system, provides details on how metadata is ingested and exposed by the system. The next section provides details on how workflows are implemented and also offers examples to help illustrate some of the architectural concepts.

4.1 Indexing System

Geospatial data collections such as satellite collections or numerical climate simulations are usually stored as a collection of files. Each of these files contain a small subset of the data, constrained by a) certain spatial or temporal ranges; b) a set of variables or parameters. Users accessing these collections often require *a priori* knowledge of how the data is structured in order to locate files of interest.

GSKY presents the abstraction of a geospatial data collection to users in a way that individual files are abstracted to end-users. Users define parameters such as the spatial and temporal ranges for their request(s) as well as the names of any requisite variables. This high level query is transformed by the indexing system into a list of files that contain parts of the data. The result can then be extracted, aggregated and presented to the user by GSKY.

To achieve this level of abstraction, GSKY relies on a heavily optimized PostGIS database, which can be queried by collection, variable names and by user-defined spatial and temporal ranges. Metadata is gathered using crawlers which periodically scan collections on local or remote file systems (or object stores such as Amazon Web Services S3), automatically extracting all the relevant information for indexing by the database. These crawlers run independently of other system components, locating new or modified files/objects to keep the database updated.

The indexing system has been designed to process and serve high volumes of metadata in near real-time. In production, the database has to be able to process queries in milliseconds, even for queries comprising large spatial areas or temporal ranges which often result in identifying, sometimes, in the order of tens of millions of files. The database has been tuned to use indexes and materialised views to achieve this level of performance. To handle any future metadata processing pressures, the database can be clustered.

[4]GEOGLAM RAPP: `http://www.geo-rapp.org/`, Production-site:`http://map.geo-rapp.org/`

4.1.1 Distributed Pipelines

At the heart of GSKY's architecture is a distributed processing system conceived as a work-flow engine written using the Go programming language. This design choice facilitates complex processes to be decomposed into a series of small generic modules which can work concurrently to produce a result (abstractly captured as a DAG). Each of these modules have a well defined input and output type and internally implement the functionality required to perform a transformation. The modules are then connected to form a network structure (a DAG) which solves a specific problem. Most of the ideas behind this architecture have being borrowed from the Flow Based Programming (FBP) (Morrison, 2010) specification. More recently, there have been efforts to model workflows in a generic way for deployment in distributed systems, such as clusters or the Cloud (Akidau et al., 2015). This approach facilitates both composability, where simple modules interact forming complex structures, and re-usability. Our proposed workflow implementation does not offer a generic execution framework for computation but rather focuses on expressing commonly used geospatial operations/idioms efficiently.

Figure 2 is an example of a workflow which generates a tile image corresponding to a WMS request. Each process in the pipeline works concurrently and communication between processes is performed using bounded queues. The queue stores the outputs of one process which are also the inputs of the next process.

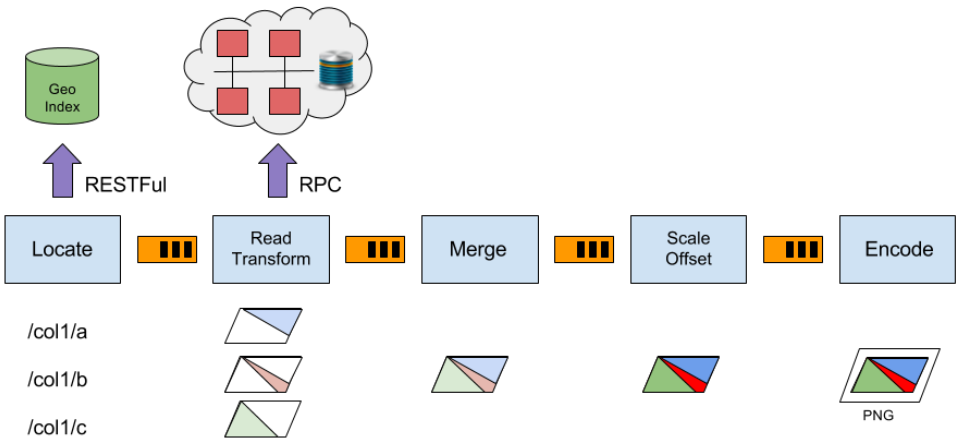

Figure 2: Sample GSKY WMS workflow handling a WMS tile request/response

Specifically, Figure 2 describes the sequence of processes to generate a tile image. The pipeline is fed by a WMS request which specifies the product, an area delimited by a bounding box and a time range. Parameters specified in the request are used by the first process of the pipeline to identify the list of files containing relevant data, using the previously described index. The second process in the pipeline is responsible for accessing each file and extracting the requested data into the right resolution and map projection. This step is the most CPU intensive and IO demanding of the whole pipeline and, in most cases, acts as the limiting factor for the pipeline. To speed up this process, work can be dynamically distributed between different nodes which operate concurrently on subsets of the files. GSKY uses a form of Remote Procedure Calls (RPC), as a way of distributing work to a cluster of remote machines over the network and collecting back the results once they have been processed. The remaining steps in the Figure describe the process of how the data is extracted from different files and merged (ie /col1/a, /col1/b and /col1/c). The final result is then scaled, corrected and encoded into a WMS compatible image format such as PNG or JPEG.

Time series analysis is a common task in geospatial analysis, where the evolution of a parameter is studied for a certain period of time. This kind of analysis can be achieved using a similar workflow to the one previously described, but exposed as a WPS.

5 Conclusions and Future Work

At the moment, GSKY can compute products on-demand. The published list of map layers and processing services is static. This is a current limitation of the underlying OGC protocols. Ideally, these interfaces should be expanded so users can dynamically specify their own operations defining a product. The new WCPS standard sets out a domain specific language that gives users the ability to specify computations combining different

products. Studying the WCPS standard and evaluating the possibility and implications of implementing it on GSKY is one of our next goals.

Another option is to work on turning GSKY into a distributed generic workflow execution engine. This would open the possibility for users to be able to define and deploy their own processes, defining custom workflows based on GSKY's underlying architecture. Having a well defined interface and general serialization protocol for the data would mean that processes could be defined in any programming language.

A near-term goal is to extend the number of datasets and services that GSKY offers.

Acknowledgements

The authors wish to acknowledge funding from the Australian Government Department of Education, through the National Collaboration Research Infrastructure Strategy (NCRIS) and the Education Investment Fund (EIF) Super Science Initiatives through the National Computational Infrastructure (NCI), Research Data Storage Infrastructure (RDSI) and Research Data Services Projects.

References

Akidau, T., R. Bradshaw, C. Chambers, S. Chernyak, R. Fernndez-Moctezuma, L. Reuven, and S. McVeety (2015). The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proceedings of the VLDB Endowment 8. 12*, 1792–1803.

Baumann, P., A. Dehmel, P. Furtado, R. Ritsch, and N. Widmann (1998). The multidimensional database system RasDaMan. *ACM Sigmod Record 27*, 575–577.

Belgun, A., K. Grochow, M. Henrikson, P. Leihn, A. Mason, M. Raghnaill, R. K., and B. Simpson-Young (2015). The Australian National Map.

Caron, J. and E. Davis (2006). UNIDATA's THREDDS data server. *22nd International Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology.*.

Cornillon, P., G. J., and S. T. (2003). OPeNDAP: Accessing data in a distributed, heterogeneous environment. *Data Science Journal. 2*, 164–174.

Deoliveira, J. (2008). GeoServer: uniting the GeoWeb and spatial data infrastructures. *Proceedings of the 10th International Conference for Spatial Data Infrastructure.*.

Evans, B., L. Wyborn, T. Pugh, C. Allen, J. Antony, K. Gohar, D. Porter, J. Smillie, C. Trenham, J. Wang, A. Ip, and G. Bell (2015). *The NCI High Performance Computing and High Performance Data Platform to Support the Analysis of Petascale Environmental Data Collections*.

Gorelick, N. (2013). Google Earth Engine. In *EGU General Assembly Conference Abstracts*, Volume 15, pp. 11997.

Guerschman, J., A. Held, R. Donohue, L. Renzullo, N. Sims, F. Kerblat, and M. Grundy (2015). The GEOGLAM Rangelands and Pasture Productivity Activity: Recent progress and future directions. In *AGU Fall Meeting Abstracts*.

Kini, A. and R. Emanuele (2014). GeoTrellis: Adding geospatial capabilities to Spark.

Lewis, A., L. Lymburner, M. B. J. Purss, B. Brooke, B. Evans, A. Ip, A. G. Dekker, J. R. Irons, S. Minchin, N. Mueller, S. Oliver, D. Roberts, B. Ryan, M. Thankappan, R. Woodcock, and L. Wyborn (2016). Rapid, high-resolution detection of environmental change over continental scales from satellite data: The Earth Observation Data Cube. *International Journal of Digital Earth 9*(1), 106–111.

Marmanis, D., M. Datcu, T. Esch, and U. Stilla (2016). Deep learning earth observation classification using imagenet pretrained networks. *IEEE Geoscience and Remote Sensing Letters 13*(1), 105–109.

Morrison, J. P. (2010). Flow-Based Programming: A new approach to application development. *CreateSpace*.

Schnase, J. L., T. J. Lee, C. A. Mattmann, C. S. Lynnes, L. Cinquini, and H. A. F. Ramirez, P. M. (2016). Big Data challenges in climate science: Improving the next-generation cyberinfrastructure. *IEEE Geoscience and Remote Sensing Magazine 4*(3), 10–22.