Manaaki Whenua
Landcare Research

A review of the current state of freely available data cube software – report for the MBIE Innovative Data Analysis programme

Prepared for: IDA Project

January 2018

A review of the current state of freely available data cube software – report for the MBIE Innovative Data Analysis programme

Contract Report: LC3377

Ben Jolly

Manaaki Whenua – Landcare Research

Reviewed by:	*Approved for release by:*
Anne-Galle Ausseil	Sam Carrick
Research Priority Area Leader	Portfolio Leader – Characterising Land Resources
Manaaki Whenua –Landcare Research	Manaaki Whenua – Landcare Research

Contents

1 Background and Introduction

Research Area 3 (RA3) of the Innovative Data Analysis (IDA) programme for the Ministry of Business, Innovation, and Employment (MBIE) led by Manaaki Whenua Landcare Research (MWLR) has the aim of integrating and/or extending a set of technologies to support a processing pipeline or workflow from data acquisition through to analysis and delivery to create an infrastructure that leverages open (web, biodata, and geospatial) standards to maximize its utility, interoperability with, and applicability to other systems. A significant component of such a system, providing centralised data storage as well as possible compute services, could be a 'data cube'.

Data cubes can be defined simply as multi-dimensional arrays (technically 'hyper-cubes' at 4+ dimensions), typically holding large amounts of data. Data cubes commonly hold spatio-temporal information such as sequences of satellite images or model output for a series of timesteps, though single snapshots (or iterations of snapshots) of non-temporal data can also be stored. This simplistic definition has been used as a basis for much more complicated solutions built to act as central data repositories for spatial and spatio-temporal information from multiple sources. These cubes are often capable of ingesting data of different types using different coordinate reference systems (CRSs) with different pixel grids and then presenting output that conforms to whatever CRS and grid is requested. This would, for example, make it possible to directly compare Sentinel-2 (European Space Agency 2018) imagery with that from Landsat 8 (U.S. Geological Survey 2013), where the data from each satellite platform have different pixel footprints (10 m x 10 m vs 30 m x 30 m native). More advanced systems also offer data discovery and browsing services that make it easier for users to find and access the data they need. Another advantage of data cubes is that they tend to store data in a way that is optimised for faster data access than a regular 'flat file' system, with some built to operate as part of a cluster that distributes analysis tasks over several computers.

The purpose of investigating a data cube for the MBIE IDA programme was to see if such technology could provide a service that would integrate data from various sources (model and observational) in a central location where it would be easily accessible to researchers. We also report on feasibility for implementation of the tested data cubes. An actual implementation would only occur if the tested data cube is deemed satisfactory, cost-effective, and fully functional.

2 Objective

To investigate different data cube options, analyse their pros and cons, and investigate potential for implementing the one most suitable for geospatial data.

3 Methods

A crucial first step was to define a set of criteria for evaluating possible data cube options, given the diversity of approaches. Analysis of existing workflows, along with discussions both within the IDA project and with other MWLR staff, highlighted important features that a final solution should provide. The most important requirement is that the data cube must be able to run on existing IT systems (GNU/Linux operating system, cluster environment). It must also be able to handle large data volumes (many terabytes of compressed data) to cover the inclusion of the MWLR satellite archive, or at least a subset of this, as well as potentially large model outputs (either intensive temporal output or to cover ensemble or Monte Carlo-style methods). Datasets at MWLR tend to span a variety of formats, including the standardised but not widely adopted 'KEA' geospatial file format (Bunting & Gillingham 2013), so it is important that any solution be flexible enough to easily ingest these formats. The final choice of data cube should be relatively easy to implement and maintain as MWLR does not want to spend significant staff resources to support a complicated or high-maintenance system. Finally, it would be good if the data cube could provide some form of data discovery service to make it easier for users to discover and use new datasets.

While MWLR is not strictly an 'open-source' shop, we tend to gravitate toward and embrace open-source technologies over proprietary counterparts. There is also currently no desire to spend large amounts of extra money on a data cube. Ideally, users should be able to interact with a cube without needing to learn new languages or having to use completely foreign tools. Common languages, environments, and tools currently in use at MWLR are Python, R, C, GDAL, Arc GIS, and QGIS. As the cube will be implemented on a cluster, it is expected that a large amount of analysis will take place within that environment. This can either happen internally within the cube system (preferable), or externally by writing code to manually extract and process data chunks (possible but less desirable). MWLR currently has access to two such compute clusters: the 'SCENZ' cluster owned by MWLR, and the 'Pan' High Performance Compute (HPC) cluster as part of the National e-Science Infrastructure (NeSI). The SCENZ cluster is architecturally similar to Pan but easier for MWLR staff to access and control so initial focus will be on implications for that system.

These requirements were distilled into the following list, in order of importance:

1 Must be able to run on GNU/Linux within MWLR's SCENZ cluster environment

2 Must be able to handle large volumes (10+ TB) of spatial data and include compression

3 Must be able to handle a variety of input data formats, CRSs, and resolutions from import through to analysis

4 Must be relatively straightforward to implement and maintain without requiring a considerable time investment

5 Must provide some form of data discovery

6 Ideally, has a sizable existing user base to increase both software longevity and the chance of getting help from the community with any issues that arise

7 Ideally, free of charge to acquire, preferably open-source

8 Ideally, be usable without extensive system-specific retraining of staff (i.e. integrate well with Python/R and provide APIs for data access from other tools).

9 Ideally, provide methods for easily extracting data to be viewed in standard GIS packages such as ArcMap or QGIS

10 Ideally, provide a distributed compute service

Given this list of criteria, the next stage was to perform a survey of available options that fulfil as many of the criteria as possible. It was intended that the top three candidates would then be tested for implementation, first on a workstation and then, if successful, on the SCENZ cluster. Test datasets would be imported into the cubes and simple benchmark tests of compute speed and data storage space would be carried out. Timeframes for implementation were restricted to approximately one week per data cube with the intention of spending an additional two weeks testing and evaluating all three options.

4 Results

4.1 Overview

Based on the criteria mentioned in Section 3, the top three candidate data cubes chosen for testing were the Open Data Cube (ODC),[1] rasdaman,[2] and SciDB.[3]

On paper, the ODC seemed to be the top option because it appeared to fulfil all requirements. It is based on the original Australian Geoscience Data Cube developed by Geoscience Australia as a framework for storing and analysing satellite data over the entire Australian continent in an HPC environment. The open-source community around the ODC is relatively young but is growing rapidly, along with development and adoption of the cube.

Conversely, rasdaman and SciDB are more established data cube solutions however they both suffer similar limitations. Both have open-source variants (for example rasdaman Community Edition or 'CE') but are essentially non-free in commercial environments or for use at larger scales due to capability restrictions in the 'free' versions.

All data cubes make some use of 'GDAL',[4] the de facto industry standard library for processing geospatial data. This means all are capable of at least reading the multitude of file formats that GDAL supports. However, exact compilation options for each specific GDAL installation are very important as the KEA file driver used by MWLR is considered as an 'optional extra' due to its reliance on the HDF5 library.

The ODC was investigated and refined following attendance at the Open Data Cube Workshop organised by the Centre for Space Science Technology (CSST) and CSIRO with

[1] https://www.opendatacube.org/ and https://github.com/opendatacube/datacube-core

[2] http://www.rasdaman.org (open-source Community Edition) and http://www.rasdaman.com (enterprise)

[3] https://www.paradigm4.com/technology/

[4] GDAL - Geospatial Data Abstraction Library (http://www.gdal.org/)

seven attendees from New Zealand CRIs and universities. Substantial improvements to the ODC were implemented between July 2017 and March 2018 – a reflection of the growing community behind it – so all information in this report has been updated to reflect the current state as of March 2018. There was not sufficient time to completely revisit rasdaman CE or SciDB; however, a brief review suggests little has changed with either of those options (at least not enough to change the findings of this investigation).

4.2 Data cube descriptions

4.2.1 The Open Data Cube

The ODC[1] is essentially a python framework with utility scripts that wraps around a database (typically PostgreSQL) as well as GDAL. Using the Dask and XArray Python libraries (along with several tools such as celery and redis[5]), the ODC can spread processing over several cores within a single machine or across an HPC centre. The intention is that this is largely hidden from the users, who interact with the cube using either Jupyter Notebooks[6] and Python code for live interactions, or regular Python scripts for batch processing.

The ODC is designed around the use of plugins that developers can build to extend the framework to cover different database engines, data storage systems (such as Amazon S3), and data services (such as WMS). Some plugins are available already, such as the S3 storage driver, however others are in various stages of development (such as WMS).

Datasets can either be *indexed* (to tell the ODC where to find the original files) or *ingested* (the ODC reads the original data files and writes the contents to a different format more optimised for processing speed). Either way, the bulk of the data remains within the file system as geospatial files with only metadata and indexing information stored in the database.

One potential disadvantage of the ODC is that adding new datasets is not yet a completely trivial process, so certain processing workflows would be better off saving results outside of the cube environment.

The biggest disadvantage of the ODC is an almost complete lack of detailed technical documentation for power users and system administrators, however this is being improved and the community is very active through their 'Slack' message board (https://opendatacube.slack.com/, join at https://opendatacube.signup.team/).

4.2.2 Rasdaman Community Edition

Rasdaman[2] CE is a more complicated 'array database management system' that optimises storage and retrieval of very large multi-dimensional geospatial arrays from the database, again relying heavily on GDAL. All data need to be ingested into the database before they

[5] http://www.celeryproject.org/ and https://redis.io/ - tools to help distribute tasks between computers

[6] http://jupyter.org/ - an interactive electronic 'notebook' with embedded code and figures

can be analysed, and the internal compute engine relies on multiple database servers to distribute the workload of a database query between themselves. This works well on systems designed for large, scaling database servers; however, much of this functionality is restricted to the commercial version.

Users interact with the data cube primarily through 'rasql', a variant of the 'SQL' database programming language for both extracting raw data from the cube as well as performing in-cube calculations (from simple NDVI indices to complex models or classifications involving multiple input datasets) and returning results. An API[7] does exist with several libraries in other languages (C, Java, Python, R, etc.), though all retrieval and processing still happens via rasql.

One very attractive feature of rasdaman CE is that it features a very comprehensive set of standards for data access such as WMS, WPS, and WCS[8] cores plus all extensions. It also supports WCPS, the Open Geospatial Consortium's raster processing language for performing in situ computation on remotely hosted data.

There is a substantial amount of rasdaman CE documentation available. However, when investigating specific questions about technical implementation on a cluster, there seemed to be missing information, perhaps encouraging users seeking that functionality to purchase the enterprise edition and the associated support.

4.2.3 SciDB

SciDB[3] is like rasdaman in that it is also an array database management system and uses its own variant of a query language, but builds off different underlying array technology and is not specifically targeted at 'raster data' or geospatial data in general. Third parties have created drivers that allow GDAL tools to retrieve and insert data from/to SciDB cubes as though they were files (https://github.com/appelmar/scidb4gdal). For example, this allows the use of the 'gdal_translate' command to easily insert data into a SciDB array; however, this process can be somewhat clumsy.

SciDB claims to have an advanced set of in-database analysis operations and very good 'massively parallel' architecture, however some of this functionality is limited in the open-source version (sometimes referred to as a 'demo version' in documentation) as it does not support any form of secure communications between the processing nodes. This makes it hard to implement on most networked systems in any kind of well-run organisation. SciDB is also designed to run on older versions of Ubuntu (12.04 or 14.04) or CentOS (6) yet still appears to be actively developed.

[7] Application Programming Interface – a way for third-party code to access functionality of a given package

[8] Web Map/Processing/Coverage Service(s) – standards for retrieving geospatial data from (and sometimes processing it on) servers or cloud systems

4.3 Data cube comparisons

Table 1 - Comparison between selected data cubes using the initial criteria

	ODC	rasdaman CE	SciDB
GNU/Linux-compatible and cluster friendly	Very, with some small tweaks	GNU/Linux-compatible, not friendly to a cluster not set up for database usage	GNU/Linux-compatible, not friendly to a cluster not set up for database usage, not friendly to corporate networks
Large volumes of spatial data and include compression	Yes	Yes but no compression	Yes
Handles a variety of input data formats, CRSs, and resolutions from import through to analysis	Yes	Yes	Not really, this is a manual process on top of SciDB
Relatively straightforward to implement and maintain without requiring considerable time investment	Partially. The easiest of the three within a 'real' (non-VM) environment. Expected to get easier	Partially (testing VM broken)	Not really
Data discovery	Yes	Yes	Yes
Free of charge, preferably open-source	Yes	Partially	Partially
Usable without extensive system-specific retraining of staff	Yes	Partially	Partially
Provides methods for easily extracting data to be viewed in standard GIS packages	In development	Yes, many	No
Ideally provide a distributed compute service	Yes, with minor tweaking (in development)	Partially but very clunky in open-source version	Yes, but not in a way that works within the MW network

Error! Reference source not found. shows a comparison of features between the ODC, r asdaman CE, and SciDB data cubes according to the criteria laid out in Section 3. The cube most suited to these criteria is the ODC, with rasdaman CE and SciDB each possessing one or more major flaws.

One of the biggest issues with rasdaman CE is its advertised lack of data compression, practically a deal-breaker when it comes to handling large-scale satellite imagery. SciDB's largest flaw is the reliance on third-party tools to provide geospatial capabilities, and it appears that the level of geospatial capability is somewhat lacking compared with rasdaman CE and the ODC.

Common problems across both data cubes are the lack of scalable compute in the open-source versions, and that, when it comes to calculations, both cubes use new variants of query languages for most of the heavy-lifting. This increases the barrier to adoption as users would need to be trained in how to interact with the data cubes. This contrasts with the ODC, which operates in a Python/GDAL environment that is more familiar to existing users, albeit with different libraries. The ODC is also much more suited to the setup of the SCENZ cluster, and is fully open-source. Currently, the biggest flaw of the ODC is the documentation, though partially implemented features come a close second.

4.4 Implementation test

Implementation of all three cube options was substantially more difficult than anticipated for various reasons. Lack of documentation was a common and significant barrier that cost significantly more time than was available, especially with substantial improvements to the ODC code and documentation made available too late in the project.

One important finding is that the open-source variants of both rasdaman (CE) and SciDB are both more limited than the front pages of their websites imply, and it was not until the 'nitty gritty' of the install/testing process was reached that some of the true implications became apparent.

The first attempt at installing the ODC on a local workstation was relatively successful after some difficulty with GDAL versions (ensuring that we had a version compiled against the HDF and KEA libraries). However, there were significant difficulties attempting to either index or ingest MWLR's satellite data into the system due to the nature of the import system, which consists of a series of potentially complicated configuration files that are completely undocumented. Additionally, it was unclear both how the distributed computing aspect functioned, how and where most of the data would be stored, and whether they would be compressed. These problems lead to the temporary abandonment of attempts in favour of rasdaman.

The rasdaman website provides a Virtual Machine (VM) of Ubuntu with rasdaman CE pre-installed for initial testing; however, portions of this were not working. VMs also provide limited value as no processing of any real data volume can take place, so exist only to prove that someone was able (partially) to get the website running. Native installation of rasdaman was not as straightforward as the documentation implied and this was increased by the fact that GDAL had to be rebuilt from scratch to support KEA files. A small-scale version was successfully installed on the one node of the SCENZ cluster; however, it quickly became clear that the distributed compute option would not be compatible with the SCENZ cluster, which is not set up to run as a database server farm. At this point the installation process was paused so SciDB could be tested.

The SciDB installation process was relatively straightforward on a local workstation using Docker[9] and images provided by the same third party that created the scidb4gdal driver mentioned above (https://github.com/appelmar/scidb-eo). Unfortunately, the SCENZ cluster is not compatible with Docker and it took time to attempt the re-implementation of the system within the image on the cluster. Ultimately, it was decided not to invest much time in this approach as even when the main part of the system was installed, it was doubtful how well it would run on the cluster due to the security requirements (or lack thereof) of the open-source version of SciDB.

At this stage of the project there were three partially working data cubes in various states of repair and progress could be described as disappointing.

The ODC attempt was briefly revisited in November 2017, during the initial drafting of this report, when a revisit of the ODC website showed substantial updates to the documentation. Other work commitments meant this was not pursued further until March 2018 where easy access to the ODC developers was provided at a 2-day workshop. Over the course of that workshop the latest version of the ODC was successfully installed on a laptop with the appropriate GDAL drivers for KEA files. It took a further day to finalise the configuration files required to describe the MWLR Sentinel 2 custom data products so they could be ingested. Documentation is still sorely lacking in this respect; however, personal contact with the ODC developers, and the community via the Slack channel, eventually managed to solve most problems. The likely amount of time and effort that would be required to finish the implementation became the main reason for discontinuing transfer of this install to the SCENZ cluster for further evaluation. Although it is possible that a working solution could be achieved within 2 weeks of full-time work, this is not guaranteed, and would likely involve further contact with ODC developers.

It should be mentioned that a fourth option for a cube exists that may satisfy many of the criteria in Section 3, GRASS GIS databases, but this had not been considered until after all other options had already failed. Reasons for ignoring this option were user unfamiliarity, potential issues operating within a cluster environment, and difficulty using third-party tools to view data contained within the database.

5 Conclusions

The current 'open-source' tools offered lack the documentation and polish required for straight-forward implementation and hassle-free maintenance. All options investigated had their shortcomings and none was an 'ideal' candidate, though the Open Data Cube (ODC) came close. This technology is not yet at the point where it can be easily implemented for large-scale data by someone who is not intimately familiar with the system, though it is likely that the process would be significantly easier if licenses for the enterprise editions of rasdaman or SciDB were purchased.

[9] https://www.docker.com/ - a 'containerisation' software platform that makes it easier to download and run complicated systems of software packages with minimal effort

It is possible that identified shortcomings of GRASS GIS could be overcome more easily than the issues identified with the other solutions. However, this would require additional time outside the scope of this report.

6 Recommendations

Technologies exist but the set-up and deployment effort is significant. GRASS GIS is a possible solution that was not seriously attempted in this project; however, it may be worth investigating in the future. There is also considerable development happening in the ODC project, so that may change soon. One option would be to engage further with one of the projects and attempt to implement something over a longer time where deadlines are less critical, thus developing MW expertise and helping shape the tool chosen into something that may be more beneficial to MW in the long term. This would, however, require strategic investment.

7 Acknowledgements

This report has been funded through the Ministry for Business, Innovation and Employment contract number C09X1412 entitled "Innovative data analysis for reporting and decision making".

8 References

Bunting P, Gillingham S 2013. The KEA image file format. Computers and Geosciences 57: 54–58. https://doi.org/10.1016/j.cageo.2013.03.025

European Space Agency 2018. SENTINEL-2 Overview. Retrieved May 31, 2018, from https://sentinel.esa.int/web/sentinel/missions/sentinel-2/overview

US Geological Survey 2013. Landsat 8: U.S. Geological Survey Fact Sheet. Fact Sheet, 3060, 3–6. https://doi.org/10.1002/0471743984.vse9497