Omar Scaling

Mark Lucas mlucas@radiantblue.com
Principal Scientist
RadiantBlue Technologies Inc.

Overview

OMAR is a web based geospatial processing system providing services, products, and browsing for the enterprise. The primary components include a workflow system, a spatially enabled relational database, and all of the processing of the OSSIM geospatial processing engine. Multiple file directories are repeatedly checked for input and new source material. The staging process creates reduced resolution data sets and histograms for subsequent viewing, metadata is parsed and inserted into the database.

Users interact with the system through a map based query system, OGC Web Mapping Services (WMS) or Service Oriented Architecture Protocols (SOAP). Value added products, precision terrain corrected ortho-rectified views, and streaming video clips are produced on demand by the system from the source data in the repositories. The OSSIM geospatial processing engine can produce a wide range of image chain products, services and views.

This paper discusses the strategies employed by OMAR for scaling. OMAR is a functioning prototype system built from mature open source components. Different usage scenarios place variable loads on the system. Optimizations of these scenarios will be described along with the internal workflows and configuration of the OMAR system.

Finally, an OMAR White paper and Roadmap are available for additional understanding of the system.

OMAR Architecture

OMAR can be installed and tested on a laptop. A typical installation would install all components on a Linux based server or PC. Larger enterprise solutions spread out the work and OMAR functions over multiple servers. The current development roadmap includes plans to add federated services for wide area distributed search and processing. Different operational scenarios stress different components of the system. The primary software components of the OMAR system are:

- Tomcat
- Apache Web Server
- Java Virtual Machine
- OpenLayers
- MapServer
- Postgres/PostGIS Relational database
- OSSIM Processing Engine
- OSSIM Staging Tools
- MPI (Parallel Processing Libraries)

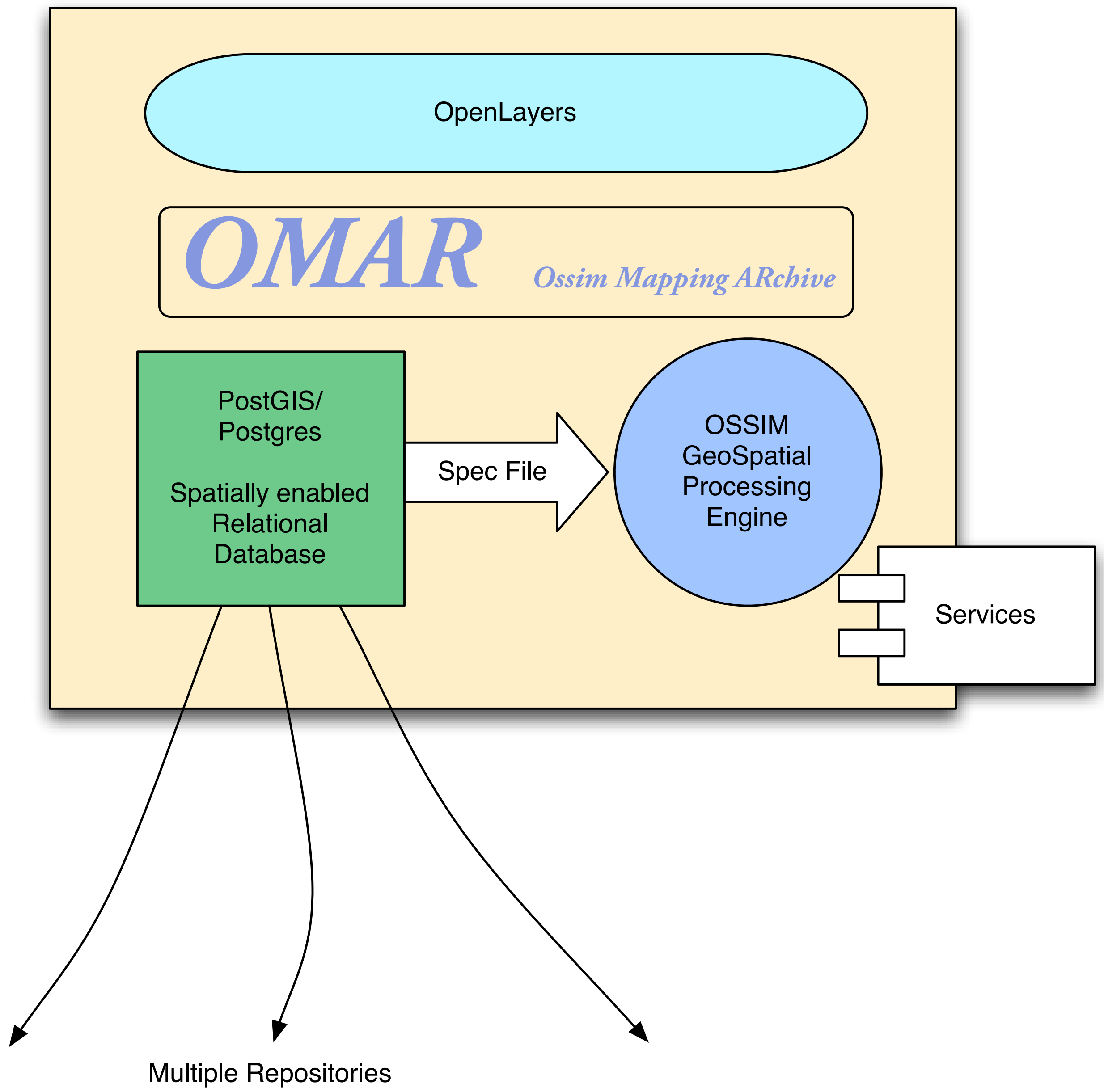

Figure 1 - OMAR Primary Components

OMAR Workflow

Multiple repositories can be scanned for new data sets. If a data set is not in the database the OSSIM staging tools build reduced resolution data sets and histograms for subsequent use. The meta data is then inserted into the relational database.

Users can query the system through a map based interface and the relational database. The results are displayed for subsequent selection and manipulation. Alternatively, OGC WMS or SOAP interfaces may be used to request data and production from the system.

These actions generate parameters that are fed into image chain templates for product generation. Raw materials in the repositories are processed on demand by the ossim geospatial processing engine to generate the requested service, view or product.

Ingest and Staging

Multiple repositories can be scanned and ingested into the OMAR system. Staging a new data set will generate reduced resolution data sets, histograms, and targeted file formats if desired. An API exists allowing these processes to be managed by an external work flow manager. The staging process can be performed in parallel on multiple input servers or can be run as additional processes on an integrated OMAR server. The various phases are described below:

Ingest Scripts

The OMAR system maintains a list of sub-directories to monitor. Internal scripts, or an external workflow management system can control the frequency of how often those repositories are scanned. The internal OMAR scripts recursively traverse the repository sub-directories looking for new data sets that are not contained in the relational database.

Reduced Resolution Sets

Reduced resolution sets or “Image Pyramids” are created where they don’t exist. These are created in a tiled Tiff format providing rapid access to various resolution levels and rapid indexing into areas of interest within the level. The OSSIM processing tools and engine the appropriate level and tiles for subsequent image chain processing. The tiled nature of the format is taken advantage of by a tile sequencer within the OSSIM baseline. Image tiles and their associated image chain transforms can be sequenced to multiple threads, processes, and processors for parallel computation.

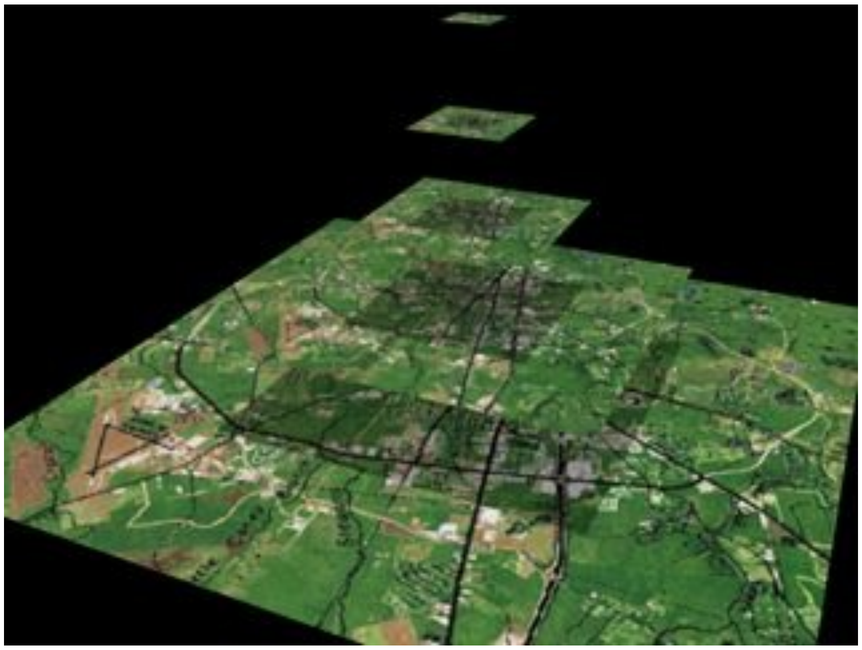

Figure 2 - Reduced Resolution Data sets

Histogram

Histogram stretching is employed by default when viewing data sets. The high dynamic range of many sensors needs to be stretched into the proper range for the human visual system. By default this requires every pixel to be scanned to build the histogram profile. For performance, this has been integrated into the sampling required for the reduced resolution set generation. Tuning within the system is also available to sub sample images or resolution levels for quick histograms. These are one time operations that are performed during the staging process so that the results can be immediately used for subsequent processing and viewing.

Meta-data ingest

The relational database stores records, pointers, and thumbnails to all of the data assets. Tools in the OSSIM baseline parse and extract the metadata from a wide range of geospatial formats. All of the meta data is stored in the database record. Commonly indexed fields are maintained for sensor type, latitude, longitude, acquisition date and time, target identifier, etc. This enables temporal, spatial and relational queries.

Database Queries

Postgres/PostGIS is the default relational database of the OMAR system. Previous government funded testing and studies have demonstrated that this outperforms the leading commercial solution in the four tested areas:

1. Ingest and Insertions
2. Indexing
3. Single User Random Spatial Queries
4. Multiple User Random Spatial Queries

Relational Queries

Postgres/PostGIS enables spatial and relational queries within the OMAR system. OpenLayers is used as a front end map viewer, MapServer is used for WMS and WFS transport. Work is currently under way to replace MapServer with the Ossim Mapping Service (OMS) which will provide higher system performance. Standard database optimization techniques need to be applied to tuning the OMAR database system. This will lead to internal schema restructuring, indexing additional fields, and some internal database normalization.

Temporal Filtering

Currently, the temporal fields and queries are developed and maintained within the OMAR logic. We have been in talks with the PostGIS developers about integrating temporal and elevation capabilities into the spatial query system.

Viewing Selected Results

Queries into the relational database return tabular selected results. The user can currently drill into the native data sets through the records.

In the case of imagery and viewer window is invoked and a WMS interface back to the native file is established. Behind the scenes the OSSIM engine is pulling pixels from the native data and resolution sets through an image chain that performs orthorectification and precision terrain correction and histogram stretching to the images in the chain.

In the case of UAV video, the video clip is converted to a flash stream and fed to the user where it is viewed through an external client with a flash plugin. Work is currently being accomplished to organize sequential clips into playlists for continuous viewing.

Both of these cases can be thought of as a stream of pixels through a processing flow. Established strategies exist for spreading this load across external servers as the number of users, processes, and streams increase.

Production

The interactive viewing of the system invokes the background processing described in the previous discussion on viewing. Services and production orders can generate much larger products that follow the same process. Production is a back end process that will need to be separately scaled as the demand for system resources increase. The architecture of the system minimizes the number of computations and the data access through a parameter based image chain processing flow.

Image Chain Processing

The OSSIM libraries, tools and production engine are built on the concept of image chain processing. The following diagram from the ImageLinker analysis tool provides a visual representation of an OSSIM image chain:

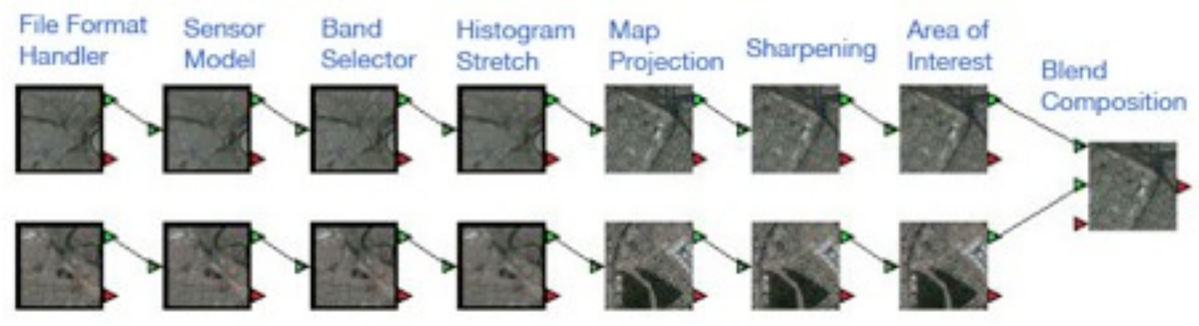

Figure 3. OSSIM Image Chain Processing, pixels in the view (right) are pulled through the image chain from source pixels on disk (left)

Image chains are non-destructive, parameter based processing flows. To create a view to the screen or a file on disk, pixels are requested to flow through a series of handlers, filters, transforms and combiners from the source pixels on disk. The specifications for these image chain templates are stored as ‘spec’ files within text files or database records. Typically each unit within the chain can be modified through its adjustable parameters. A detailed discussion of image chains is beyond the scope of this paper. For scaling purposes, image chains can be simply thought of as pixels being pulled through mathematics to produce results in a file or within a view.

This type of processing falls into the ‘embarrassing parallel’ category of computations. A divide and conquer approach can be taken to tile the pixels from different areas of the image and independently send them to different processors to run through the image chain. OSSIM incorporates OpenMPI for parallel processing and a tiling sequencer for this purpose.

Elevation Sampling and Precision Terrain Correction

By default, images are map projected and precision terrain corrected by the system. The OSSIM libraries automatically index DTED, SRTM or Raster elevation sources to retrieve the appropriate elevation cells underlying the projected image. Pixels are shifted appropriately through supported rigorous sensor models or Rational Polynomial Coefficients (RPCs) for the associated data sets.

Ortho Rectification

The default image chain processing map projects the source data to a geographic projection. Other map projections are optionally available. The data is resampled and pixel values are remapped as the histogram stretch is applied. These are CPU bound calculations that occur with image viewing and product generation.

OMAR Stress Scenarios

Load on the system is generated in two primary areas:

1) A large number of users querying and viewing with the system

2) Product generation from staging, on demand orthos, services and products

In the simplest terms, this can be segregated into front end and back end loading of the system. Standard techniques for scaling and replicating and optimizing the front end can be applied to the architecture to address scaling the number users, queries and interactive viewing. The back end load requires parallel processing, tiling sequencers, MPI and scheduling systems that are used commonly in super computing calculations.

Front end Loading

For front end loading we anticipate patterns of a large number of users simultaneously accessing hot spots and large numbers of users accessing different data sets. Standard scaling techniques for cacheing, server replication, and load distribution can be employed. On the Large Data infrastructure we are manually configuring this scaling strategy. In the future we need to integrate cloud computing approaches.

Ingest Load

External workflow management is currently being used to manage multiple front end servers for the ingest and staging process. The current strategy is a simple round robin job queue tasking dedicated ingest servers.

Cloud Computing

Large Data is NOT cloud computing. Large Data is a High Performance Distributed File System. This is cloud computing explained in just over 5 minutes, it is worth a look:

http://www.youtube.com/watch?v=QJncFirhjPg

OMAR can be hosted in a cloud environment. Typically, OMAR is hosted on one machine, for web server, database, etc. This is fine few very low traffic but we definitely need to explore something else if we are to run in a production environment. On Large Data we have distributed the ingest and processors across multiple servers. The two critical elements I see are basically caching, and load balancing.

Caching:

We need a cache for more ortho-rectified, precision terrain corrected tiles. We all know, but have a hard time convincing our customers of this, that is is an expensive operation. We've been able to do a few "tricks" to speed things up for Reference Imagery (Blue Marble) by using TileCache from MetaCarta. But, this is only good for static imagery. Just as with MapServer's WMS GetCapabilities, this is an external config file and every time a new image is added, you'd have to add it to the config file. This is not a viable solution for a dynamic back end OMAR generates "on the fly" just like we did for OMAR's WMS. What we need is a "smart" cache will track how often or when the last time a tile was accessed in the cache. If a tile becomes stale, or outlives its usefulness, it can be purged from the cache to free up resources. If it is needed again, it can always be recomputed and place back into the cache. This should save quite a few CPU cycles as it reduces an expensive operation to just a simple HTTP GET if the tile in question has been requested recently.

OMAR can apply squid or memcached which will manage resources such as files. It will keep certain ones in memory if it is a "hot commodity".

Load Balancing (or Scalability and Availability):

This is where the "cloud computing" approach (or Grid, Cluster, etc) comes in. We have begun initial investigation into this area but need more research. Simply buying more hardware is not the practical answer since the Virtualization approach has come along. Now the answer is to employ a high performance machine and run a cluster of VMs on it to simulate hardware. We develop/test Linux OMAR already with VMWare. We are evaluating delivering future instances and upgrades as Virtual Machines in those environments that support it. This makes sense as everything is already installed and configured as a "Gold Standard" package that can be readily cloned and distributed for either placement on the server, or even deployed to a laptop in the field will all of OSSIM/OMAR/Planet/ImageLinker/etc all set up and ready to go.

We are evaluating an OpenSource project called TerraCotta (www.terracotta.org). What this does is basically turn a machines on the network (physical or virtual, just need an IP) in to one large JVM. It handles all the load balancing for you and when you launch a process in a thread, it will pick the most available resource to execute. Nodes can come and go and it will handle fail over. A short video is worth reviewing:

http://www.youtube.com/watch?v=miM5-Kvh6jE

We've talked to some of the TerraCotta folks back at SpringOne in December and attended a session by Ari Zilka.

Streaming Viewers (JPIP)

The OMAR viewer makes efficient use of the available bandwidth by only calculating and providing the pixels in the view. The pixels are uncompressed jpeg to the remote browser or to any OGC WMS compliant client.

J2K JPIP streaming is a protocol that requires a JPIP compliant client. The OSSIM library has support for the JPEG 2000 format with the Kakadu libraries. The Kakadu library is generally accepted as the leading implementation for J2K and JPIP streaming. The OMAR roadmap contains a task to implement JPIP streaming. This will require that J2K Images are created as part of the staging process or as a back ground task under the control of the user. As previously mentioned, a JPIP client will be required to support OMAR JPIP streaming.

Flash Conversion and Playback

The OMAR system ingests Motion Imagery Standards Board (MISB) compliant UAV video clips. The stream is processed for the embedded metadata for positioning, date, and acquisition time. Upon selection by the user, the MPEG clips are converted to flash for streaming playback in an external streaming application.

Back end Loading

The dynamic production capability of the OMAR system provides a different set of scaling challenges. Resampling, projection, terrain correction and histogram stretching is applied as images are output from the system. These functions occur through the OSSIM libraries which contains OpenMPI support. Emphasis has been placed on multithreading, tile sequencing, and MPI parallelism to scale this work across available hardware.

Product Generation

OMAR is distinguished from other web based systems with its dynamic back end processing. Raw materials in the repositories is sampled, projected, and transformed through the image chain to the desired view or product. These operations are typically CPU bound with double precision mathematical calculations. As more pixels are pulled through the image chain the load on the backend system will rise in a linear fashion. This demand can get generated by an increasing number of users or the generation of large products for delivery.

Elevation Processing

The current implementation traverses DTED elevation trees to select the correct cells for topographic adjustment. This approach requires traversal of sub-directory structures and the open and closing of a large number of DTED files. Because of this additional processing elevation adjustment has generated a delay of a couple of seconds for interactive views.

A number of optimizations to this approach are currently being investigated. Preprocessing elevation data into an optimal format and memory mapping elevations will substantially increase performance in this area.

Parallel Processing Libraries

OSSIM has integrated support for MPI parallel processing. Additionally, OpenMPI on the Lustre file system has been optimized to take advantage of the low latency infiniband fabric. The OSSIM libraries contain a tile sequencer that can be tuned to parse out sequences of image tiles for image chain processing to slave processors. Previous profiling have demonstrated almost linear scaling with this approach.

Scheduling Systems

More sophisticated control over the back end system can be accomplished with a job scheduling system. Large jobs can be prioritized and scheduled with these systems.

Summary

It is important to note that OMAR is a prototype system that is in very active development. Implementations are currently being demonstrated and tested in government and academic environments. Various tools, and strategies exist that will allow the system to scale in the future.

Front end scaling is available through standard cache mechanisms used in high demand web systems. The ingest system can be easily scaled across multiple ingest servers.

The back end geospatial production system requires tuning of OpenMPI based tile sequencers within the OSSIM library

Finally, cloud replication services needs to be looked at and evaluated to minimize the amount of parallel process tuning for future implementations.