A NEW INITIATIVE FOR TILING, STITCHING AND PROCESSING GEOSPATIAL BIG DATA IN DISTRIBUTED COMPUTING ENVIRONMENTS

A. Olasz [a]*, B. Nguyen Thai [b], D. Kristóf [a]

[a] Department of Geoinformation, Institute of Geodesy, Cartography and Remote Sensing (FÖMI),5. Bosnyák sqr. Budapest, 1149 (olasz.angela, kristof.daniel)@fomi.hu
[b] Department of Cartography and Geoinformatics, Eötvös Loránd University (ELTE), 1/A Pázmány Péter sétány, Budapest, 1117 Hungary, ntb@inf.elte.hu

Commission ICWG IV/II

KEYWORDS: Distributed computing, GIS processing, raster data tiling, data assimilation, remote sensing data analysis, geospatial big data, spatial big data

ABSTRACT

Within recent years, several new approaches and solutions for Big Data processing have been developed. The Geospatial world is still facing the lack of well-established distributed processing solutions tailored to the amount and heterogeneity of geodata, especially when fast data processing is a must. The goal of such systems is to improve processing time by distributing data transparently across processing (and/or storage) nodes. These types of methodology are based on the concept of divide and conquer. Nevertheless, in the context of geospatial processing, most of the distributed computing frameworks have important limitations regarding both data distribution and data partitioning methods. Moreover, flexibility and expendability for handling various data types (often in binary formats) are also strongly required.
This paper presents a concept for tiling, stitching and processing of big geospatial data. The system is based on the IQLib concept (https://github.com/posseidon/IQLib/) developed in the frame of the IQmulus EU FP7 research and development project (http://www.iqmulus.eu). The data distribution framework has no limitations on programming language environment and can execute scripts (and workflows) written in different development frameworks (e.g. Python, R or C#). It is capable of processing raster, vector and point cloud data. The above-mentioned prototype is presented through a case study dealing with country-wide processing of raster imagery. Further investigations on algorithmic and implementation details are in focus for the near future.

1 INTRODUCTION

Our goal is to find a solution for Geoprocessing of big geospatial data in a distributed ecosystem providing an environment to run algorithms, services, processing modules without any limitations on implementation programming language as well as data partitioning strategies and distribution among computational nodes. As a first step we would like to focus on (i) data decomposition and (ii) distributed processing. The challenges associated with each focus area, related methodology and first results are analyzed and discussed in the paper.

2 PROBLEM DESCRIPTION

2.1 Geospatial Big Data

Reliable analysis of the geospatial data is extremely important base for being able to support better decision making with location-aware data even in our changing World. The challenges for handling geospatial big data include capture, storage, search, sharing, transfer, analysis, and visualization (Jewell et al., 2014). Furthermore, with newly adapted data management requirements and initiatives, even more open data will be available on the web which need to be handled, the latent information be shared and extracted knowledge applied in the level of decision making as well (Wu and Beng, 2014.). Big data, open data and open government has joint interest in location and in many challenges considered in geospatial aspect will soon benefit from it (Jewell et al., 2014). We consider GI analysis as a principal capability in a way to transform information to knowledge.
The progress and innovation is no longer hindered by the ability to collect data. The most important issue is how we exploit these geospatial big data (Lee and Kang, 2015). We consider that, we are facing the paradigm shift from data-driven research to knowledge-driven scientific method in Big Data which was considered as a challenge by R. Kitchin in 2014. In our previous work (Nguyen Thai and Olasz, 2015) the Big Data concept and the well-known four dimensions: *Volume, Velocity,* Variety (originally Laney came up with this three dimensions in 2001) and *Veracity* have been discussed taking into account of geospatial considerations and characteristics. Additional dimensions have been continuously appearing to describe better the concept of the workflow such as: Value, Validity, Visibility, etc. (Li et al, 2015).
Moreover, we have concluded a fifth *"V"* to Geospatial Big Data as *Visualization*, because it has a great importance to communicate the desired information in Geographical Information Science from the very beginning. From geo-location *information* transform into *knowledge* going through the pipeline of Data Life Cycle in every phase of conversion (collect, aggregate, analyse, return as knowledge, share) *Visualization* is fundamental (Nguyen Thai Binh and Olasz, 2015). Beyond, we consider that in Big Data environment *Visualization* plays an important role in (1) geospatial data processing (due to volume and variety) that help analysts to identify trends, relations, correlations and patterns in an efficient way; (2) facilitate broadcasting geo-information to citizens (and decision makers) employing interactive and eye tracking approaches. According to our conclusion *Visualization* is a definite geospatial component of Big Data.

2.2 Defining Geospatial Big Data

Spatial data (also known as geospatial data, geo-information, geodata, etc) have many definitions depending from the background of the author. All of them emphasize the geographic location of the phenomena to be described as basic criteria. The nature of the digital representation of the continuous space can be grouped in 4 or 5 type. Traditionally we consider two type of geospatial data vector and raster (Elek, 2006) owing to the development of information technology nowadays we can have higher abstraction type of data such as point clouds, graph networks. An additional particular kind of location-aware data is also examined by analysts; social media-like data which requires a particular approach to collect and process as well. Along with Big Data theory *geospatial big data* is defined as volume, variety and update frequency rate that exceed the capability of spatial computing technology (Lee and Kang, 2015, Li et al., 2015, Kambatla et al., 2014). In Table 1. we have collected the main characteristics of geospatial big data for each type of formats such as: representation formats, GIS operations, volume, velocity, variety and visualization aspects. To have a better understanding on what are the main attributes of geospatial data because it is hard to delineate the margin starting to *"exceed the capability of spatial computing technology*". To estimate the size of the processable amount of data are use-case specific, there are some good examples (Evans et al., 2014) where the authors tried to identify the Geospatial Data and Geospatial Big Data differences.

Data type	Formats	GIS operations	Volume	Velocity	Variety	Visualization
Vector	point, line, polygon (multi)	Overlapping vector geoprocessing	available amount of vector data (for instance nation-wide cadastre, or land cover,roads, waterways, utility network, etc)	real time monitoring, and rapid response is emerging	consider in a GIS processing several types of previously mentioned data need to be combined to extract the relevant information	thanks to OGC standards and web-gis platforms non researchers are able to use GIS data for many different purposes
3D representation	point cloud,TIN (or Triangular mesh)	3D modeling, urban modeling,simulation,flight from above camera view, visibility operations, semi-automatic point cloud feature detection, classification, terrestrial laser scanning, BIM	available amount of point cloud data (or TIN) for the creation of DSM, DTM modelling, feature extraction and simulation requires huge computational capacity	time sensitive 3D data requires rapid processing (disaster management and simulation)		3D view (perspective) together with thematic content with reduced information are essential to spreading information for different level of end-users
Raster	grid	Local, Focal, Zonal (Global) Map Algebra processing, image analysis	available (free) series terrestrial, aerial and satellite (multispectral and hyperspectral) imagery (airplane, UAV), earth observation data requires huge computational capacity using raster image processing methods	real time spatio temporal earth observation data processing is need more than ever (independently from the extent of the processing)		to deliver results of earth observation monitoring and processing novel solution in visualization also needed to transform information human readable
Network	graph (nodes, edges), line	routing, network analysis, allocation (Geo-business)	trillions of edges, nodes for graph processing available from location based networks (also from social media) originally big data concept made for text-based and graph-like structures	real time monitoring of moving objects, transportation decision support is needed		in network analysis and routing visualization techniques are indispensable to serve it real time
Geolocation-aware media	text:post, tweets, web-logs,check-ins, media: GPS tracks from smart phones,UAV video,geoPDF profiles: name, geocodes,	disaster management decision support, crowd sourcing, human geography, sociology, crime mapping uses geoprocessing, GEOINT techniques	data mining, geostatistical techniques and predictive modelling are traditionally considered as big data processing methods which requires computing capacity even on web content analysis (text based media files)	real time social media and information flow is faster than ever, geospatial sector is already taking part		for location-based social media visualization is the basis which exceed the traditional barriers of GIS sector, novel solutions are rising every day to collect and analysing geolocated web content.

Table 1. Geospatial Big Data characteristics

According to the previously mentioned definitions and characteristics of Big Data and Geospatial Big Data represented in the table are reasonable. We do not intend to identify other characteristics of Big Data, because they are not closely related to Geoprocessing topic. In order to process Big Data distributed computing environment and techniques have been introduced and applied to handle time-consuming operations. In our related work feasibility aspects were targeted together with a conceptual framework for benchmarking experimental processing.

3 PREVIOUS AND RELATED WORKS

3.1 Distributed computing

Distributed computing environment is a software system where computational and storage components are on networked computers ,communicating and coordinate their actions by passing messages through "network socket" endpoints within the network. Components interact with each other to achieve a common goal. Three significant characteristics of distributed systems are: concurrency of

components, lack of a global clock, and independent failure of components. Plainly, a distributed system is a collection of computers within the same network, working together as one larger computer. Massive computational power and storage capacity have been gained due to this architecture. We must note that processes' running in a distributed system does not share memory with each other, like parallel computing systems. Processes in distributed system communicate through message queues. Two architectural models are suggested for distributed computing systems:

- Client-Server model: where clients initiate communication or processing job(s) to the server, which distribute that request(s) to all processing and storage units if necessary to do the real work and returning results to client.
- Peer-to-Peer model: where all units involved in distributed system are the client and server at the same time, without any distinction between client or server processes.

The technology of used in distributed computing of geospatial data is similar to any other process of distributed computing. Several solutions are introduced to accelerate geoprocessing usually time consuming methods. Along with the available amount of data together with its particular procedures to derive information geographical information systems constantly invokes novel solutions from the IT sector.

3.2 Distributed geospatial computing

The Encyclopaedia of GIS (Phil Yang 2008,) defines distributed geospatial computing (DGC) as *"geospatial computing that resides on multiple computers connected through computer networks". So "geospatial computing communicates through wrapping applications, such as web server and web browser"*. To translate it, distributed geospatial computing is when geoprocessing is done within a distributed computing environment. In the Handbook of Geoinformatics (2009) Yu et al. focus on the multi-agent system with ontology to perform distributed processing of geospatial data. The distributed processing of geospatial data is continuously evolving together with the evaluation of computer networks. In this study due to limited length, the introduction of the historical evolution of DGC is not in purpose. A single milestone we would like to emphasize from the evaluation progress is when Google Earth was issued in 2004; by reason of it caused a life changing effect on the citizens' everyday life and made popular geospatial applications. Furthermore, until nowadays Google's solutions are leading in the process of massive dataset along with the development of easy-to-use interface (e.g., Google BigTable) and play an important role in the open source community developments. Consequently, distributed systems supports heterogeneous network and infrastructural background, cloud solutions have been developed to exploit the advantages of distributed systems and made available services for geospatial computing as well.

3.3 Cloud computing

Cloud computing involves deploying groups of remote servers and software networks that allow different kinds of data sources be uploaded for real time processing to generate computing results without the need to store processed data on the cloud, while focusing on maximizing the effectiveness of the shared resources. As we see cloud computing system has evolved from distributed systems (Bashar, 2013).

Plainly, a cloud is aiming for scalability and flexibility providing computational and storage power to customers and users (Mei et al., 2008). Main components are:

- Clients (any end device having Internet connection can be a client)
- Data-center (server farm)
- Distributed servers

The key factor in cloud computing is the power of virtualization by creating virtual machines on-demand. The model of the cloud computing allows access to handy resources like: storage, applications, computing capacity and services in an efficient way (Jajodia et al., 2014). Even some data provider and cloud infrastructure provider cooperated and incorporate advanced services with available public data sets (for example over 85 TB of Landsat 8 satellite imagery, or NASA NEX images of the Earth's surface available on Amazon AWS).

3.3.1 Cloud computing model

Often cloud computing model demonstrated as a stack of computing services. At the bottom of the stack Infrastructure-as-a-Service (IaaS) take place, symbolizing the basis of the system. (Sosinsky, 2011). In the middle of the stack the Platform-as-a-Service (PaaS) sits providing the tools designed to deploying applications dynamic and powerful. On the top of the stack Software-as-a-Service (SaaS) facing the customers with the software and Application Programming Interfaces (APIs).

Infrastructure-as-a-Service (IaaS)

The user accesses the storage, network and basic computing resources, can deploy and execute particular programs, the operating system. The user is free from the management of the infrastructure background but has right to configure the operating systems, storage, deployed applications, and perhaps narrow control of select networking components (e.g., host firewalls). The user's operating system together with applications can be migrated to the cloud and substitute the data center of the company.

Platform-as-a-Service (PaaS)

The user can install or develop own web software on the virtualized environment that are created using programming languages and tools supported by the provider. The user doesn't have right to make changes on operating system, storage, network and servers only to the deployed apps along maybe with application hosting configurations.

Software-as-a-Service (SaaS)

The users can use the predefined applications on the cloud accessible with heterogeneous devices through web browser. The user can only configure the software but the underlying solutions are hidden.

Data-as-a-Service (DaaS) and Data-as-a-Product (DaaP)

The users have access to dataset (public or private made available on the cloud) and probably the software

environment to process the data as a service on the top of the hidden architecture. The user not allowed making infrastructural, network or storage parameterisation. Within such centralized solution the data management (frequent update, cleaning, conversions) are served by the provider (Olson, 2009). Data-as-a-Product concept defined by Huang (2015) is a *"small sized summary of the original data and can directly answer user's queries".*

In the next section we are introducing a comparison matrix for current technologies from the Geospatial aspects of the usage. In the Table 2. our intention was to collect aspect from the advanced GIS users who would use the provided services on their own data without programming knowledge. Also some case studies were collected as instance to harvest information on practical realization. Also wanted to gather information on data management flexibility in decomposition, association and distribution and control of the data distribution among nodes but it was tough to dig deep in implementation details of each tool. We admit that these environments are not only expensive to build, but they require highly-trained DevOps Engineers to maintain them and grow them as the data accumulates. At the popular ranking site of DB-Engines we can derive information on the evolution of popularity calculated with a complex method of scoring (Fig.1.). Method of calculating the scores of the DB-Engines Ranking: based on the popularity of the system (number of mentions of the system on website, searches in Google Trends, number of interested users on the Stack Overflow and DBA Stack Exchange, number of offers on the leading job search engines Indeed and Simply Hired, Number of profiles in professional networks, in which the system is mentioned LinkedIn, number of Twitter tweets, in which the system is mentioned (db-engines.com, 2016). The most popular database engine which is highly excels is HBase from those 8 engine, the second is SPARK SQL and Accumolo right away following it. Rasdaman is lacking behind since the publication but the trend is rising.

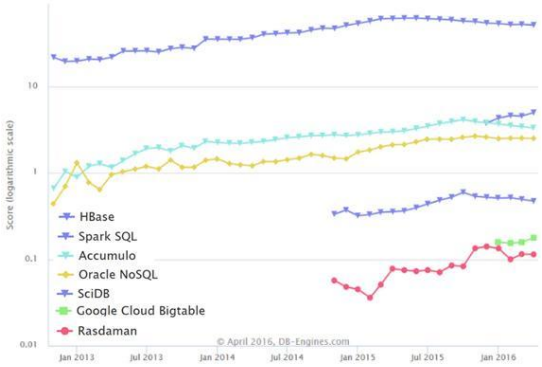

Fig.1. Comparison of the DB engines according to DB-Engines Ranking from April 2016 (1.HBase, 2. Spark, 3. Accumulo, 4. Oracle NoSQL, 5. SciDB, 6. Google Cloud BigTable, 7. Rasdaman (Hadoop, GeoTrellis, Akka, Geowave, GeoMesa are not DB Engines to be compared)

	FILE EcoSYSTEM	FRAMEWORK	TOOLKIT	DB-ENGINE
Name	Apache Hadoop (+Spatial Hadoop)	Apache Spark (+GeoSpark)	Akka (+GeoTrellis)	Google Cloud BigTable (*commercial)
Short Description	allows distributed processing. Spatial Hadoop is MapReduce framework for spatial data	fast and general engine for large-scale data processing	for building highly concurrent, distributed, and resilient message-driven applications on (JVM)	managing structured data that is designed to scale to a very large size
Supported GIS data type Input/Output	Point, Rectangle and Polygon	Point, Rectangle, and Polygon	no GIS alone, together with GeoTrellis Raster	together with GeoMesa, Google's geo related apps.
GIS case studies/projects	Calvalus system, several projects GIS functions,mainly operating with vector data	Research projects writing GIS functions on the top of Spark	Clarus, GeoTrellis, see more with GeoTrellis below listed.	nearly all of Google's largest applications, GeoQlick
Supported GIS processing (or executable languages)	no GIS alone, together with Spatial Hadoop or other modules vector processing is supported, or other predefined solution	Python, R, (java) no GIS alone, together with developed services capable predefined GIS functions,	no GIS alone, together with GeoTrellis, Java	-
Operates over HDFS	YES			
Database Model	File Server	RDD	Actor Based Model	Wide column store (NoSQL)
Execution of existing processes written in any language	Java	Java, Scala, Python, R.	.NET Framework, Mono via the Akka.NET project.	Java, Go
Data Management flexibility	Not supported	yes, utilizing Spark Core	Advanced Scheduling	Sharding
Scalability potential	Scalable	Highly scalable	Scalable	Incredible scalable
Programming paradigm (MapReduce)	Supported	Shared memory	Sharding raster data across the cluster	Supported
Supported OS /Platform dependencies	Linux, Unix	Linux, OS X, Windows	cross-platform (Java Virtual Machine)	hosted
Development programming language	Java, C, shell scripts, Pigeon	Scala,	Developed in Scala and Java	Java, Python, Go, Ruby

Table 2. Technical comparison matrix

In the following table we have collected the most popular frameworks supporting distributed computing with GIS data. We wanted to investigate the capabilities of each framework, which data types are supported or suitable for that particular framework, what kind of infrastructure are they supporting, what kind of data storage are they using (Table 3).

	DB-ENGINE	FRAMEWORK	DB-ENGINE	DB-ENGINE	DB-ENGINE	TOOLKIT	LIBRARY	DB-ENGINE
Name	Apache HBASE	GeoTrellis	SCiDB	Rasdaman	Apache Accumulo	GeoMesa	GeoWave	Oracle Spatial and Graph Georaster
Short Description	open-source, distributed, versioned, modelled after Google's Bigtable	Scala framework for fast, parallel processing of geospatial data.	Array database designed for multidimensional data management and analytics	Allows storing and querying massive multi-dimensional arrays	based on Google's BigTable design and is built on top of Apache Hadoop, Zookeeper, and Thrift.	open-source, distributed, spatio-temporal database built on cloud data storage systems. highly parallelized indexing strategy	library for storage, index, and search of multidimensional data	delivers advanced spatial and graph analytic capabilities to supported Big Data platforms
Supported GIS data type Input /output	no GIS alone together with predefined programs vector, point	supports vector, raster formats and services ,also supports raster data processing on Apache Spark together with GeoTools	Using GDAL to convert GIS data in a format of SciDB	together with Petascope supports raster formats and OGC standards, to import GDAL is used	no GIS alone	vector, raster together with Spark, Geoserver, GeoTools and GDAL	vector, raster, grid together with Spark, Geoserver, GeoTools and GDAL	vector, raster, point cloud, TIN, OGC web services, RDF semantic graph, independent from any file formats
GIS Case studies/projects	research papers mention applications developed on the top of HBase for GIS	Model My watershed, City of Asheville, Priority Places, Visualizing Emancipation, sea level rise,urban forest	several studies (mainly using raster processing) using R, IDL, Scala, R	Earthserver, PublicaMundi Project	together with geoMesa and GeoWave	some example studies	Accurate Geology Reproduces Observations, Microsoft Research GeoLife page	wide use cases and project from situational analysis, cloud computing, cartography, network and 3D applications, etc.
Supported GIS processing (executable languages)	predefined gis actions developed by third party	predefined GIS actions Map Algebra like, local, focal,zonal,global,statistical operations	predefined gis actions developed by third party	python, R, QGIS, and ESRI (SQL-style query language)	predefined gis actions developed by third party	predefined gis actions	predefined gis actions	Groovy-based console to execute Java and Tinkerpop Gremlin APIs complex gis operations.
OPERATES OVER HDFS	YES	YES	NO	NO	YES	NO	YES	YES
Database Model	Wide-column store based concepts of BigTable	RDD (Resilient Distributed Dataset)	ARRAY DATABASE (Multivalue DBMS)	multi-dimensional array db	key-value store, Wide column store	Key-value store, (on Accumolo, KafkaDB)	Key-value store, (on Accumolo)	enterprise grid computing technology
Execution of existing processes written in any language	C,C#,C++,Groovy,Java,PHP, Python,Scala	pre-programmed	R, Python, MatLab,IDL, C++ and SAS style	R, rasql, GDAL (image processing library)	Java	R	pre-programmed	Groovy and Python
Data Management flexibility	Immediate Consistency	no information	not supported	not supported	no information	RDD transformations, Filters, Aggregations, Distributed Raster Computation	Hadoop MapReduce for bulk computation	Data security, replication, partitioning, bulk load utilities are readily available.
Scalability potential	scalable, default	default	yes	no	yes	yes	yes	GeoRaster object can flexibly have one devoted RDT table to improve scalability
Programming paradigm (MapReduce)	Sharding	Sharding raster data across the cluster	Array (shared nothing architecture)	Array shared nothing and shared-disk	highly scalable structured store based on Google's BigTable	Using MapReduce, Accumulo, Spark, Storm, Kafka, Cassandra	Using MapReduce, Accumulo, Spark, Kafka	Parallel processing capabilities are added into mosaicking, pyramiding and all raster algebra functions
Supported OS/Platform dependencies	Linux, Unix	Linux, Mac OSX, NSF, Windows with rsync	Ubuntu 12.04, CentOS 6, Red Hat Enterprise Linux (RHEL) 6	Linux SuSE 9.1 / 9.2	Linux,Centos 6.6 Java7Hadoop 2.2.0,.ZooKeeper 3.4.x	Accumulo,Hadoop, Zookeeper, or Kafka, Java JRE or JDK, Apache Maven,on Linux	Linux	Linux, Windows
Development language	Java	Scala	AQL	C++	Java	Scala	Scala	Tinkerpop Blueprints and Rexter REST APIs, and Java graph APIs
Structure (Modular, Components)	Bigtable-like capabilities on top of Hadoop and HDFS.	Modular	no moduls	mixed applying OGC standards	COMPONENTS: TabletServers, one Garbage Collector process, one Master server and many Clients.	Modular	Modular	Components

Table 3. Technical details of existing solutions on distributed GIS processing

5 DESIGN AND IMPLEMENTATION DETAILS

IQLib has been initiated by three project partners in IQmulus project, namely FÖMI-Hungary, IMATI-Italy and IGN-France. During the first two years of development, we have experienced that some of our services are not compatible with Hadoop infrastructure including HDFS file system. While most of current processing frameworks follow the same methodology as Hadoop and utilize the same data storage concept as HDFS. One of the biggest disadvantage from processing point of view was the data partitioning mechanism performed by HDFS file system and distributed processing programming model. In most cases we would like to have full control over our data partitioning and distribution mechanism. Also most of our services cannot be altered to fulfil MapReduce programming model, due to the nature and logic behind those services. In this paper we are focusing on the data distribution Tiling and Stitching and Data Catalogue components of the conceptual framework called IQLib. On the Fig 2 the architectural concept is described our future intention is to create a framework for distributed processing of R, Python or any other language written geospatial analysis.

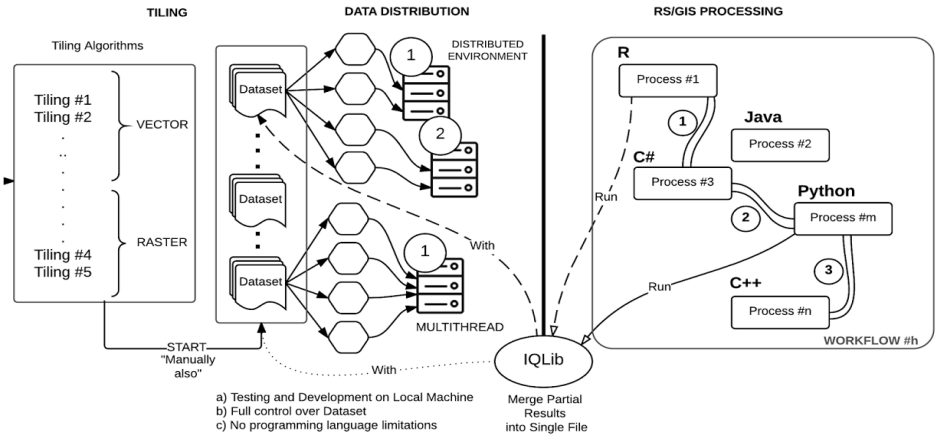

Fig.2 High level concept of IQLib processing framework

5.1 Introduction of IQLib

At the beginning we have created a feasibility study on technological and conceptual aspects. The outcome of this was presented in our previous paper, where we have described the architecture of this demo application as well as processing results on calculating NDVI values using Spot5 satellite images for Hungary. The processing results seemed really convincing, so we have started to design and implement IQLib framework. This framework should be able store metadata information on our dataset, tracking partitioned data, their location, partitioning method. It should distribute data to processing nodes as well as deploying existing processing services on processing nodes and execute them in parallel.

As a result IQLib has three major modules; each module is responsible for a major step in GIS data processing. The first module is called Data Catalogue, second module is Tiling and Stitching, the last module is called Processing.

Data catalogue module is responsible for storing metadata corresponding to survey areas. A survey area contains all the dataset that are logically related to inspection area, regardless of their data format and purpose of usage. We would like to store all the available, known and useful information on those data for processing.

Tiling and Stitching module does exactly what its name defines. Usually tiling algorithms are performed on raw datasets before running a specific processing service on given data. Stitching usually runs after processing services have successfully done their job. Tiling algorithms usually process raw data, after these tiled data are distributed across processing nodes by data distribution component. Metadata of tiled dataset are registered into Data Catalogue. With this step we always know the parents of tiled data. Distributed processing module is responsible for running processing services on tiled dataset.

5.2 Data Catalogue

Based on user and developer needs, we have collected those requirements and created a data model specification document.

The first version of this documentation contains the specification of data structures on how we would like to store our metadata accessible by both users and processing services. In the beginning we intended to use Geonetwork have as a metadata storage, however as we progressed on defining and refining our data model, it became clear for us that we have to implement our own metadata store.

5.3 Data model

In order to organize our data, we have gathered the most common use cases on data processing along with existing Tiling algorithms, as a result the following terms: “Survey Area”, “Dataset” and “Data file” have been introduced. A Survey Area has at least one dataset, depending on processing requests. Each dataset has at least one data file, which may vary from size to format.

Depending on the type and content of each data files, each processing service accesses them with different strategies, these strategies are called data access patterns (DAP). Data access patterns are based on local (L), global (G), focal (F), zonal (Z) and the combination of zonal and focal (ZF) - applied as a combination of topological and coordinate neighbourhood operations. Along with data access patterns, we have also categorized data into three groups: Meshes (M), Point clouds (P) and Raster data (R). Table shows the relationship between data access patterns and file groups:

DAP	File Type	Description
L	MPR	each sample is processed independently
F	M	requires access to the vertex attributes within a given edge-distance N
F	R	squared centred window in raster for e.g. convolutions
Z	P	requires all data within a given 1D range around one coordinate of the output vertex
Z	MPR	requires all data within a given a more general range around the output vertex, (e.g., 1m spherical neighbourhood) or requires all vertices within an identical attribute.
G	MPR	Take the whole file and process it.

Table 4. Relationship between data access patterns and file groups

We also store the status of each data file, whether they are "raw", "processed", "tiled" or "buffer-zone". Tiled and buffer zones will be explained in the next section.

5.4 Technical solution

From technical point of view a data catalogue should be platform independent and available for most clients. The most convenient solution was to implement a web application which provides RESTFUL web service endpoints, where client applications may connect to and utilize its functionalities. The web application itself should be as simple as possible, meaning that it should be easy to install, maintain and use.

After examining mainstream programming languages and corresponding frameworks, we have decided to use Java programming language and Spring MVC framework to implement the Data catalogue module. As for data storage we have considered using either graph database or traditional relational databases. Data catalogue will migrate from relational database to graph database when Tiling and Stitching module is completed, until then we are using PostgreSQL as data storage media (Fig 3.).

The web application itself can be deployed on almost any Java application servers. For simplicity and testing purposes, we recommend the usage of Tomcat as a default application server.

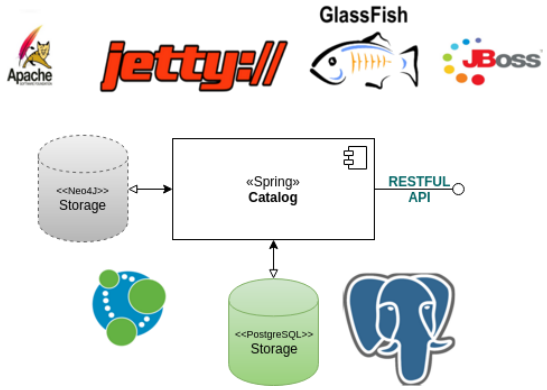

Fig.3. Architecture of Data Catalogue module

5.5 Tiling & stitching

Tiling and stitching is a composite module. In order to distributed process a large dataset, processing services require their input data to be tiled before processing. We must also think of use cases where intermediate or end results should be stitched together. This step is done by stiching algorithms; they are closely related to tiling algorithm. That is why tiling and stitching as a module should prepare and distribute data for processing services. Based on preliminary user's and processing service developer's requirements we have come up with the following data structures to model tiling aspects.

First of all we would like to define the data structure for a Tile. A Tile is a single read-write connected area of the original dataset. But from processing aspects a set of Tiles does not contain all the necessary information on its neighbours, surrounding areas. Therefore we have defined the data structure for Buffer zones. A buffer zone is a collection of data around a tile, which can be used in read-only mode by a processing algorithm to edit/process the tile (under processing) related to this buffer zone. A buffer zone can be empty; when processing algorithm does not require any surrounding data around the Tile (e.g. the service algorithm has a local data access pattern). At last, A tile and a buffer zone together creates a logical entity, called Patch. A Patch may contain two components: a Tile and related Buffer Zone depending on Tiling algorithm. As defined in Buffer Zone, there are some cases where we do not need Buffer Zones; therefore a "Simple" Patch has a Tile, where a "Compound" Patch contains a Tile and a Buffer Zone.

A tiling algorithm creates Tiles and buffer zones and corresponding Patches. IQLib tiling and stitching module should provide a set of predefined tiling algorithms developed by IQmulus project partners for service developers. As well as supporting third party developers to join and add their own tiling algorithms to the community.

After original dataset have been tiled, newly created Patches should be registered into Data catalogue module and distributed among processing nodes. Currently Tiling and Stitching module only support distribution of data via SMB and SFTP protocols.

5.6 Technical Solution

Tiling and stitching module should also be platform independent like the Data catalogue module. Our intension was not to setup any programming language limitations on Tiling and stitching algorithm developers, so that they could use the language they are familiar with (Fig 4.).

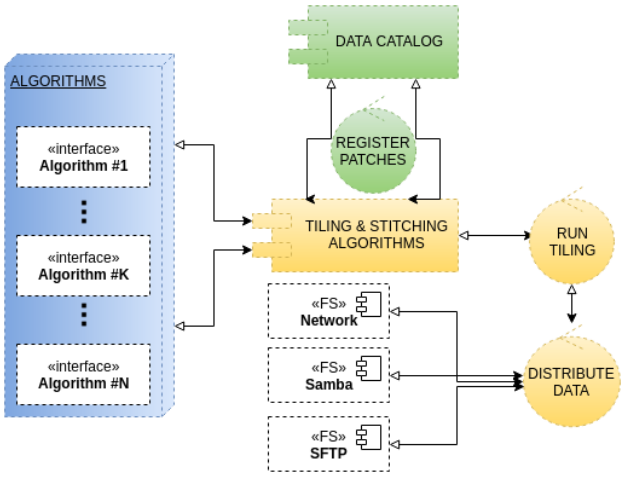

Fig.4 Architecture of Tiling and Stitching module

5.7 Processing

Partners within IQmulus project, have existing services that they would like to run on a distributed system with Tiled dataset. Currently Processing module is still under design phase with the following requirements: all services should be easily deployable and maintainable on processing nodes, each processing node should be able to inform users that currently which services are available and running, logging and notification system should be developed for both error and status report (Fig 5.).

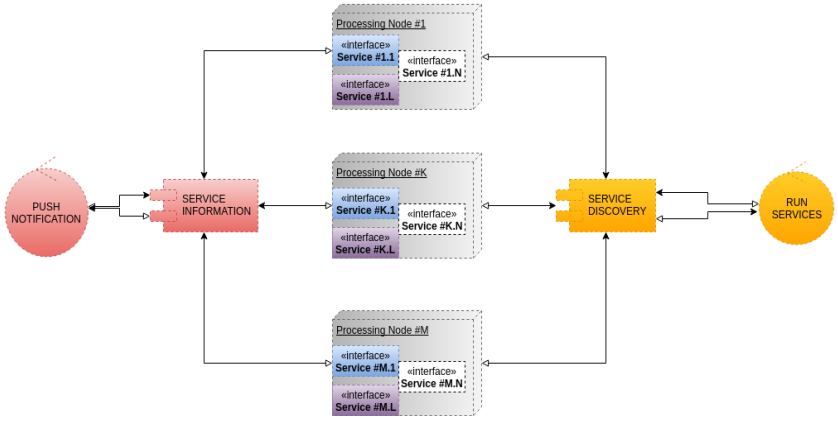

Fig.5 Architecture of Processing module

6 CONCLUSIONS AND FUTURE WORK

IQLib documentation on data model and Data catalogue is available on Github at https://github.com/posseidon/iqlib. We have decided not to publish Data catalogue module's source code until it has been reviewed and finalized by IQmulus project partners. However, the RESTFUL API is available on Heroku cloud infrastructure for all project partners to test and give feedbacks and suggestions at http://iqlib.herokuapp.com. In our future work we are going to focus on further development of the framework development along with the processing executables and experimental benchmarking of processing time.

ACKNOWLEDGEMENTS

This research is co-funded by the project "IQmulus" (A High-volume Fusion and Analysis Platform for Geospatial Point Clouds, Coverages and Volumetric Data Sets) funded from the 7th Framework Programme of the European Commission, call identifier FP7-ICT-2011-8. The research was also supported by the Hungarian Institute of Geodesy, Cartography and Remote Sensing (FOMI. We would like to thank Michela Spagnuolo research director of Institute for Applied Mathematics and Information Technologies (CNR-IMATI) for her huge effort on making opportunities as well as support on IQLib.

REFERENCIES

Elek I. 2006. Bevezetés a geoinformatikába (Introduction to geoinformatics) ELTE Eötvös Kiadó, Budapest, pp. 22-40,

Nguyen Thai B. and Olasz A. 2015. Raster data partitioning for support distributed GIS processing, Proceedings of ISPRS.Vol. XL-3/W3, pp. 543-550.

Kambatla K., Giorgos K., Vipin K., and Ananth G. 2014. Trends in Big Data Analytics. Journal of Parallel and Distributed Computing 74 (7) pp. 2561–2573.

Li S. , Dragicevic S., Anton F., M. Sester, S. Winter, A. Coltekin, C. Pettit, B. Jiang, J. Haworth, A. Stein, and Cheng T. 2015. Geospatial Big Data Handling Theory and Methods: A Review and Research Challenges, pp. 2-19.

Jewell D. et al. 2014. IBM RedBook, Performance and Capacity Implications for Big Data, IBM Corp. pp. 7-20

Laney D. 2001. 3D Data Management: Controlling Data Volume, Velocity, and Variety. Application Delivery Strategies, pp. 1-4.

Lee J.-G. and Kang M. 2015. Geospatial Big Data: Challenges and Opportunities, Big Data Research, vol. 2, no. 2, pp. 74–81.

Karimi H. A. 2014. Big Data Techniques and Technologies in Geoinformatics. Taylor & Francis Group, LLC, pp. 149-153.

Kitchin R. 2014. Big Data, new epistemologies and paradigm shifts, Big Data & Society, Vol. April–June pp. 1–12.

Eldawy A. and Mokbel M. F. 2015. The Era of Big Spatial Data. In Proceedings of the International Workshop of Cloud Data Management CloudDM co-located with IEEE ICDE

Wu, Z. and Ooi Beng C. 2014.From Big Data to Data Science: A Multi-Disciplinary Perspective. Big Data Research, Special Issue on Scalable Computing for Big Data, pp.1-4.

Yang, Chaowei Phil. 2008. Distributed Geospatial Computing (DGC). Springer US. in Encyclopaedia of GIS Shekar S, and Xiong H. ed. pp 246-247.

Bashar S. , D. A. T., 2013. Distributed Systems And Cloud Computing Systems, Computation may someday be organized as a public utility.

Mei L., Chan W.K., Tse T.H. 2008. A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues, IEEE Asia-Pacific Services Computing Conference, pp. 464-469.

Jajodia S., Krishna K., Pierangela S., Anoop S., Vipin S., and Wang C. , eds. 2014. Secure Cloud Computing. New York, NY: Springer New York.pp. 23-34.

Sosinsky B. 2011. Cloud Computing Bible. Wiley Publishing Inc.

Olson J. A. 2009. Data as a Service: Are We in the Clouds? Journal of Map & Geography Libraries, doi:10.1080/15420350903432739.

Huang G., J. He, C. H. Chi, W. Zhou and Y. Zhang, 2015. A Data as a Product Model for Future Consumption of Big Stream Data in Clouds, Services Computing (SCC), 2015 IEEE International Conference on, New York, NY, pp. 256-263.