RASTER DATA PARTITIONING FOR SUPPORTING DISTRIBUTED GIS PROCESSING

Binh Nguyen Thai [a], Angéla Olasz [b]

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

KEY WORDS: GIS, GIS data decompositioning, Distributed processing, Geospatial data association, Linked dataset

ABSTRACT:

In the geospatial sector big data concept also has already impact. Several studies facing originally computer science techniques applied in GIS processing of huge amount of geospatial data. In other research studies geospatial data is considered as it were always been big data (Lee and Kang, 2015). Nevertheless, we can prove data acquisition methods have been improved substantially not only the amount, but the resolution of raw data in spectral, spatial and temporal aspects as well. A significant portion of big data is geospatial data, and the size of such data is growing rapidly at least by 20% every year (Dasgupta, 2013). The produced increasing volume of raw data, in different format, representation and purpose the wealth of information derived from this data sets represents only valuable results. However, the computing capability and processing speed rather tackle with limitations, even if semi-automatic or automatic procedures are aimed on complex geospatial data (Kristóf et al., 2014). In late times, distributed computing has reached many interdisciplinary areas of computer science inclusive of remote sensing and geographic information processing approaches. Cloud computing even more requires appropriate processing algorithms to be distributed and handle geospatial big data. Map-Reduce programming model and distributed file systems have proven their capabilities to process non GIS big data. But sometimes it's inconvenient or inefficient to re-write existing algorithms to Map-Reduce programming model, also GIS data can not be partitioned as text-based data by line or by bytes. Hence, we would like to find an alternative solution for data partitioning, data distribution and execution of existing algorithms without rewriting or with only minor modifications. This paper focuses on technical overview of currently available distributed computing environments, as well as GIS data (raster data) partitioning, distribution and distributed processing of GIS algorithms. A proof of concept implementation have been made for raster data partitioning, distribution and processing. The first results on performance have been compared against commercial software ERDAS IMAGINE 2011 and 2014. Partitioning methods heavily depend on application areas, therefore we may consider data partitioning as a preprocessing step before applying processing services on data. As a proof of concept we have implemented a simple tile-based partitioning method splitting an image into smaller grids (NxM tiles) and comparing the processing time to existing methods by NDVI calculation. The concept is demonstrated using own development open source processing framework.

1. INTRODUCTION - 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. (D. Jewell et al., 2014). Furthermore, with newly adapted data management requirements and initiatives even more open data will appear 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. Big data, open data and open government has joint interest in location and in many challanges considered in geospatial aspect will soon benefit from it. The trend to larger data sets is due to the additional information that can be derived from analysis of a single large set of related data, compared to separate smaller sets with the same total amount of data (D. Jewell et al., 2014). We consider GI analysis as a principal capability in a way to transform information to knowledge (Fig 1.)

In Geoscience we are aware of data collection and acquire techniques which huge amount of data as introduced before, we are able to aggregate and report the extracted information from the datasets by the applied analysis now we are facing how to return it as applied knowledge to decision support and share the knowledge to everyone in an interactive or dynamic way. The progress and innovation is no longer hindered by the ability to collect data.

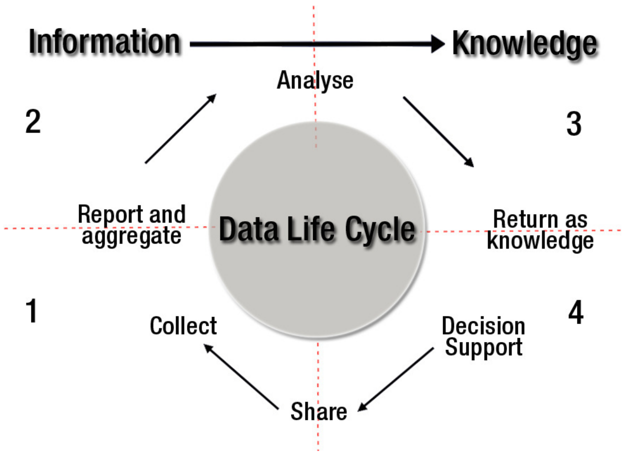

Figure 1: Data Life Cycle in the aspect of transform information to knowledge (According to R. Piechowski, 2010)

The most important issue is how we exploit these geospatial big data (Lee and Kang, 2015).

1.1 4 + 1 Dimension: Visualization

One of the major applications of future generation parallel and distributed systems is in big data analytics (Kambatla et al., 2014).

Unstructured, distributed and complexity are even more characterization of the big data sets than before (Wu and Chin, 2014). Four or as we consider five dimensions represents the big data term: (i) the Volume of information need to be treated; (ii) the Variety in the range of data types and sources; (iii) the Veracity of the source data and results; (iv) the Velocity of the processing and the grow of information; (v) (D. Jewell et al., 2014) and finally the Visualization especially in GIS world must be emphasized as a consequence of the first four "V". The Visualization of the results is significant in a Geospatial Big Data application if we compare to text based Big Data processing. GI (Geographic Information) visualization has great importance in the correct interpretation of the data, to provide easy information to non experts of GIS in order to empower citizens in location-aware issues. Hence, speed of visualization geospatial big data, the applied geospatial information design may require a great deal of effort. Rather we need to preprocess and review the incoming data on the fly for real time support of decision making than create stock on local machines or in data warehouse and later analyze them in batch. (Lee and Kang, 2015). Visualization and velocity are highly connected in geospatial big data processing.

1.2 Dimension: Velocity

The processing of geospatial big data requires computation and storage time. Currently scalability factor on storage level is not substantially relevant comparing to computation time, due to the fact that capacity of hard drives, speed and price are reducing gradually, therefore the demand of extendability on storage level of a GIS processing system can be satisfied. Whereas the need to reduce processing time has been and still is a very crucial factor in GIS processing even more in Big Data term. For modeling and visualizing of geospatially enabled contents the high computation capacity is demanding faster than ever besides the exponential increase of geospatial big data. Even though, it is far the availability of fully exploit high volume or high velocity gathered spatial data as a consequence of the finite processing performance (Lee and Kang, 2015).

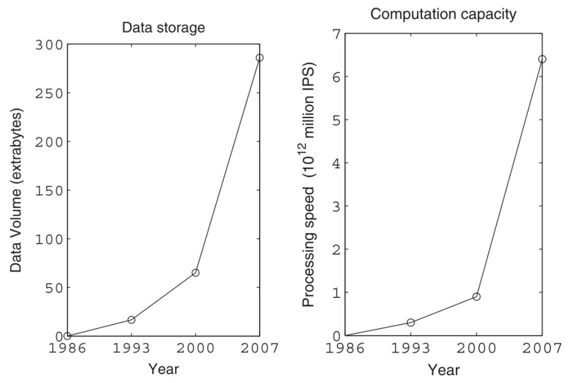

Figure 2: The increase of data size has surpassed the capabilities of computation (C.L. F. Chen and C. Y. Zhang 2014 according to M. Hilbert and P. López, 2011)

From the late nineties the generated and stored data volume has overtaken the computation capacity in the aspect of the processing speed (Fig 2.). Since then the applied techniques for fast data processing are lagging behind the volume of geospatial data to be stored. In Geoscience, introduction and implementation of new techniques to extract knowledge for non GIS experts from massive volume of data are urgent, as the existing tools are unable to process Big Data that grow so large and complex.

1.3 Dimension: Volume

The management of big geospatial data differ from other big data types. For further repeatable investigations and spatio-temporal aspect we need to store data frequently. For ease the transmission of spatial-temporal data on web map services we need to create tiles, pyramids as well the data sometimes duplicated in different formats. In order to find the appropriate dataset we also need to create, retain and serve metadata for spatial big data. Metadata creation and service incorporate already elaborated and complicated activity for the growing amount of spatial data. The storage is becoming cheaper day by day although incapable to scale up for this increasing volume. Actual answers are lossless compression and level up the abstraction of high resolution data. Another disposal is "linked to data over Web which consists of tuples of data sets which are contextualised, thereby adding value to the individual data sets" (Dasgupta, 2013). Geospatial big data are not easily linkable as text based and therefore not visible even if they are online on internet. Studies focusing on linked geospatial data are in the beginning (K. Janowicz et al., 2012) which go beyond sample data stores to model complex workflows and more than linking location to an event. If we look at the well known figure (Fig. 3) of Linked Open Data (LOD) Cloud from 2011 a very narrow sector tagged by Geographic denomination (with yellow color) even if we go in detail and search for complex geospatial informatic solution we can find some dataprovider and cloud workspace provider but most of them is "only" facts with location data collections. To be part of this model does not require to provide high level geospatial processing framework or datasets (informative anyway for the location enabled content).

This image shows datasets that have been published in Linked Data format which is "a term used to describe a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF (Resource Description Framework")", by contributors to the Linking Open Data community project and other individuals and organisations. It is based on metadata collected and curated by contributors to the Data Hub as well as on metadata extracted from a crawl of the Linked Data web conducted in 2011 and April 2014 (A. Jentzsch and R.Cyganiak, 2014). In 2014, compared to previous stage we can observe taper of the geographic sector and GeoNames has been enlarged. Today, by the realization of Spatial Data Infrastructures (SDI) as a standard provider for publishing, querying, retrieving, and accessing geodata via Web services. Additionally, SDIs offer notification and processing services and, thus, go beyond simple data stores (K. Janowicz et al., 2012). However, a large percentage of the data might not be of interest in a particular task. Facilities for filtering and compressing in order of magnitude are getting more important. Intelligent filtering and weighting without discarding data samples are challenges in a volume of big geospatial data. (D. Jewell et al., 2014).

1.4 Dimension: Variety

Data availability have been improved a lot, due to technological and policy advancements, but raw data itself does not represent knowledge or information on their own, therefore information and knowledge extraction have been applied on these data with different smart solutions to deal with the amount and variety of data. As mentioned before now we are in the third quadrant in the Data Life Cycle (Fig. 1.) the transition information to knowledge, the collection and access different kind of data is no longer a problem. We are struggling with the variety of input data (different in format, scale, source, accuracy, resolution, acquiring technology, date, purpose, processing environment, etc.)

to be preprocessed and analysed in order to provide different data outputs. Also need to be concerned combining a wide range of data formats that are commonly not located on the same platform, in the same institution and are continuously changing. "These unique circumstances resulting several performance and capacity challenges. Most big data projects face more challenges from variety and fewer from data volumes" (D. Jewell et al., 2014).

1.5 Dimension: Veracity

Veracity cope with uncertanity or imprecise data. According IBM redbook veracity also means in infrastructure level "To address the performance and capacity challenges that arise from lack of veracity, it is important to have data quality strategies and tools as part of a big data infrastructure (D. Jewell et al., 2014). To avoid false, fraudulent, or unauthorized data in a system security also need to be take into account by authorization and secure technology (even in a level of governmental level).

We need to consider that all from four or five "Vs" has great impact in GIS but besides the Volume of the raw data to be processed in the fastest way (Velocity) others are only being introduced as a consideration in GIS solutions or only in the process of being required by a GI system, such as the Veracity of the GIS datasets. However, it is important to have data quality strategies and tools as part of a geospatial big data infrastructure (D. Jewell et al., 2014), not to mention the security aspects for the data access to avoid unreliable results. This situation poses huge challenges on traditional data processing applications and data management tools in Geographic Information Science and Systems (Lee and Kang, 2015). Now we are facing the problem of the spatial big data processing in the aspects of volume and variety but later on after new solitons in processing spreaded out the GIS and related community will face soon the other and new V's challenges such as Variability and Value or rapid Visualization.

1.6 Current technologies

Distributed frameworks have been introduced during recent years like Hadoop (Apache), Spark (Apache), Akka (Typesafe) or Disco (Nokia), however these frameworks are focusing on distributed processing of text-based data and heavily rely on Map-Reduce programming model and distributed file storage. In GIS processing world, Map-Reduce programming model combining with distributed file systems like Hadoop distributed file system is not a silver bullet to solve computational tasks. The main reason behind it is that distributed file systems split data into smaller chunks, these chunks are replicated and spread among slave nodes, however we have very little control over this step and splitting mechanism based on size limitation. As a result data splitting method will invalidate the GIS input data.

1.7 Goal

Processing of geospatial big data can be time consuming and difficult. Depends on highly heterogeneous data type and processing methods different aspects need to be considered like speed, precision and real-time. 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. The data decomposition and the NDVI calculation were tested using Landsat 8 imagery for the territory of Hungary with a ground resolution of 30 m.

2. TERMS: DISTRIBUTED AND CLOUD COMPUTING

2.1 Distributed computing

Distributed system is a software system where coputational and storage components are located networked computers communicating and coordinate their actions by passing messages through "network socket" endpoints within a 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 archicture. 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.

2.2 Cloud computing

"Cloud computing" or put it simply "Cloud" 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 (**?**).

Plainly, a cloud is aiming for scalability and flexibility providing computational and storage power to customersusers. It's major components are:

- Clients (any end device having internet connection can be a client)
- Datacenter (server farm)
- Distributed servers

The key factor in cloud computing is the power if virtualization by creating virtual machines on demand

3. TECHNICAL SURVEY

3.1 Exiting distributed computing frameworks

By evaluating existing software components and frameworks, like

Hadoop and HDFS
Core Hadoop distributed computing system based on Map-Reduce and distributed data file system (HDFS).

ESRI Hadoop
ESRI solution for handling mainly Vector based data in Hadoop environment, by converting vector data into GeoJSON format, then store them on HDFS.

Spatial Hadoop
It supports Hadoop Map-Reduce programming model as well as HDFS.

Hadoop Image processing interface
HIPI is an image processing library designed to be used with the Apache Hadoop MapReduce parallel programming framework.

Akka - Typesafe
A distributed, concurent framework and runtime environment. Applications are deployed and distributed automatically by the framework.

Disco - Nokia
Distributed computing framework in Python using Map-Reduce and distributed file system developed by Nokia.

Map-Reduce Geo
Map-Reduce Geo, geospatial toolkit designed to provide raster based geospatial capabilities that can be performed at scale. However there is lack of documentation or tutorial on how to use it.

3.2 Technical limitations

In overall, the above mentioned frameworks, support working with both vector and raster based data. However, Map-Reduce programming model must be used, in order to process data and all the APIs are written in Java language, which leads to several questions:

1. What if we do not want to re-write our algorithms and services to Map-Reduce programming model, but would like to use the advantages of distributed computing environment? Is it possible to do so, with small amount of modification on existing code?
2. What if we would like to implement new algorithms and services, by using another language then Java?
3. What if we would like to controll the sequence of data distribution among the nodes?
4. What if we would like to implement our own data partitioning methods, depending on processing algorithm's nature?
5. What if we would like set the way of data distribution to storage nodes?
6. What if we would like to send functions, modules or classes to process data, without writing a wrapper application for every function?
7. What if we would like to run pre-installed executables on processing nodes without parameterizing mapper and reduces executables?

3.3 Requirements

Based on these questions, we are able to build the following set of system requirements:

1. Executing existing programs on distributed environment.
2. Implementing new GIS processing services on any language and execute them on distributed environment.
3. Full control over data partitioning and distribution.
4. Free from Map-Reduce programming model.

In order to fulfil these requirements, we need to look for an alternative distributed computing solution and the right programming language to implement the "proof of concept" processing system.

3.4 Choosing the right programming language

Before jumping into implementation, a study has been made on which programming language should be used based on the following criteria:

- Platform dependencies.
- Support on GIS libraries and data types.
- Ease-of-use and descriptive syntax.
- Well documented and community support.

Choosing the right implementation language for processing GIS data is a delicate matter. On GIS library support, popular third generation languages like C# or Java, the community support and documentation for GIS libraries is not really good. Whereas script languages like Python or Ruby are keeping up with most of the GIS related programming libraries. However, third generation languages provides several distributed computing frameworks, unlike script languages.

For example, almost any modern language is suitable for processing vector based data, it does not require a lot of technical knowledge to use them. Working with raster based data are more complicated because of the size and variety of data, as well as the background knowledge developers should gain before beginning to implement any processing algorithm.

Documentation pages are well maintained in case of python and ruby, we can not say the same thing for Java or C#. Community support and activity is also better in case of script languages than third generation languages.

Indicators shows that scripting languages are the most suitable for implementing GIS algorithms and services. Github.info and Redmonk statistics shows that Python is more active and used by more developers than Ruby. In the end we have chosen to use Python as the primary implementation language to be used.

3.5 Distributed computing framework - Dispy

So far, we have selected the programming language to work with, which is Python. Next, to find a library or framework supporting distributed computing in Python. Only one suitable candidate is available currently, based on development activity and community support, namely: **Dispy**.

Dispy is a distributed and parallel computing framework for or with Python based on **asyncoro** which is Python library for asynchronous, concurrent, distributed programming. Some of it's features are:

- Computations (both Python and standalone, meaning executables) and their depencies like files, Python methods, classes or even modules are distributed among all the computing nodes automatically.
- Computation nodes can be on local or remote network.
- Supports SSL communication.
- Ready made Job Scheduler.
- Initialized Jobs have callback functions. This feature can usefull, when we want to execute a sequence of jobsin batch.

In the next section, we will discuss some of the data decomposition and distribution methods.

4. DATA MANAGEMENT

In order to achieve distributed processing, data must be available among all processing units. Mainstream distributed file systems like HDFS (Hadoop distributed file system) or DDFS (Disco distributed filesystem) has the following main functionalities:

1. Automatically chunk data by size.
2. Distribute data across storage nodes.
3. Replicate the same data across storage nodes.

However, distributed file systems like Hadoop distributed file system (HDFS) does not provide a library or any service to support data partitioning and distribution, because all these functions are automatically done by HDFS in a transparent way, thus we have no control over. HDFS have been designed to process large text-based data, which can be easily cut and chunked by line or size. For GIS input data like GeoTIFF, we can not apply cut by file or by size, because the partitioned files will be corrupted. Work arounds exists for storing these kind of files on HDFS, like reading the header section of the original unpartitioned file, then read the partitioned data chunks to process them, but, it's still a work around, not an elegant solution. GeoTIFF represents a small segment of raster data types, therefore other binary based raster data wiill face the same limitaions. Limitation of HDFS over data partitioning on GeoTIFF data leads us to think over and considering on writing our own data partitioning and distribution module, namely a data management module.

To implement a data management module, as the first step we need to decompose data ,then distribute them on processing units. Replication functionality is not relevant at this stage, therefore we are not dealing with it. The method for data distribution may vary on the nature of processing algorithms, not to mention, sometimes processing algorithms requires information regarding topologically related data during processing phase, therefore relations, associations between original source data and decomposed data should be well defined and searchable at any time.

4.1 Data catalog

To keep track of original files and their partitioned data, associations as well as additional information, we need to have a data catalog module. A data catalog should be simple to use and reliable. It should able to store spatial data, descriptive data and association data. It should be also able to supply stored data to processing applications. Data catalog module will be implemented in the future, for this paper let's assume that it's available.

4.2 Data decomposition

To process raster data in distributed environment data decomposition is crucial. In distributed computing environment access data in a certain location, decompose to smaller chunks ('sub-raster blocks') to achieve optimal fast processing, collect the result of the algorithms from processing units and rebuild resulting data. Many approaches exist with different pattern of coordination and communication between the processing steps and results. The main aspect is how to decompose input raster to smaller partitions. A number of alternative methods exist to raster data partitioning as well. Single pixel algorithms are processed for each pixel in isolation. Defined extent algorithms are processed for each pixel in a context of a pre-determined number of surrounding pixels. Region-based algorithms are considered to apply in a geographical application where a greater neighboring area has significance where the study area is grouped into homogeneous segment or stratum. The processing algorithms and parameters are designed for every homogeneous strata. This stratum considered as a landscape unit which is homogeneous in aspect of physical geographical factors, or any targeted application area.

One of the most trivial way to decompose data is to spatially split the original data into smaller data chunks, which can be processed independently on processing units. However decomposition methods are tightly related to processing methods and the number of processing units available in the distributed environment.

As a proof of concept, we have developed a GeoTIFF decomposer application using GDAL library. Decomposer takes a source file or directory, looking for GeoTIFF files and decompose them into smaller partitions. It takes two parameters, namely a predefined grid size (N x N) and number of processing units available in the distributed computing system. The number of processing unit is needed later for data distribution. After decomposition have been successfully performed, decomposed data are being uploaded to computation nodes.

In future terms, we would like to implement other partitioning methods based on the following conditions:

1. spatial relations: data belonging to a partition are spatially related.
2. logical relations: data belonging to a partition are logically related.
3. administrative relations: data belonging to a partition are related to an administrative unit.

To test our data decomposition application, we have selected 36 Landsat 8 images including NIR and VIS bands, covering the area of Hungary. Each of them is approximately 128Mb, in sum 4.6 GB of data. We have measured the time of data decomposition for 36 Landsat 8 images into 2 x 2, 3 x 3, 4 x 4 and 6 x 6 grids on a Dual-Core PC with 4GB of RAM:

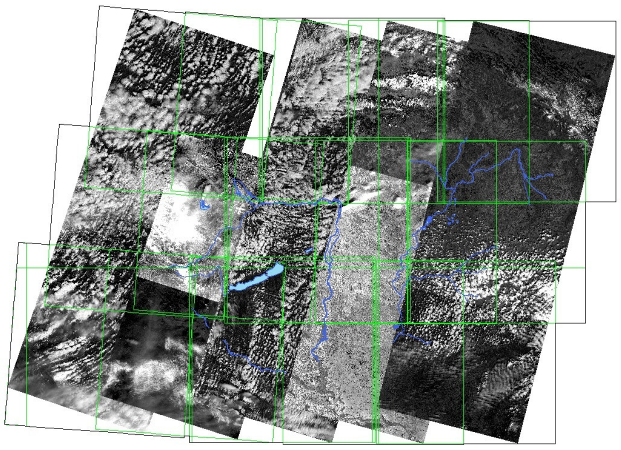

Figure 3: Bounding boxes of Landsat 8 images (band 4) of Hungary

Grid size	Decomposition time (seconds)
2×2	187 seconds
3×3	189 seconds
4×4	193 seconds
6×6	215 seconds

We have realized that, between grid size 2 and 3, there is no difference in decomposition time, however, after increasing the number of grids to 5 x 5 decomposition time have increased substantially. In future development, we would like to see, what happens, when we join all the bands into a single large file, and perform decomposition on that particular file.

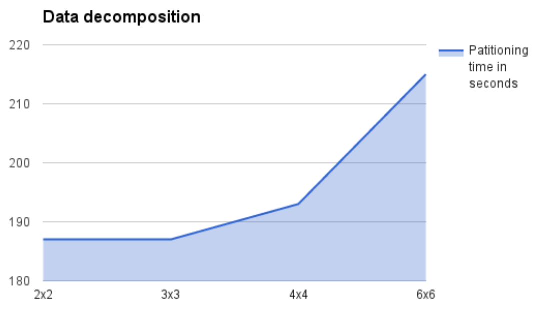

Figure 4: Data decomposition time increases after 4x4 grid size

4.3 Data distribution and re-distribution

After data have been decomposed, the next step is to distribute them among processing units. We have required the number of processing units in the beginning of the data decomposition step, thus with the knowledge of existing processing units, we have to choose how we would like to distribute partitions. The easiest method is to distribute partitions equally on all processing units. However in case of decomposition methods based on spatial, logical or administrative relations, we should distribute partitions on the same processing unit or processing units within the same network sub domain, of course this type of distribution depends on the processing algorithm which iterates from one region to another. As for the demo application, we have distributed data evenly on every processing units.

Let's take a simple example, where data have been decomposed based on administrative relation as described on Figure 3 - P_n is the unique identifier for each processing unit. In generally, processing algorithms may require spatially or topologically related regions on the same processing unit for faster processing. In order to achieve this, we need to have full control and access over data distribution mechanism.

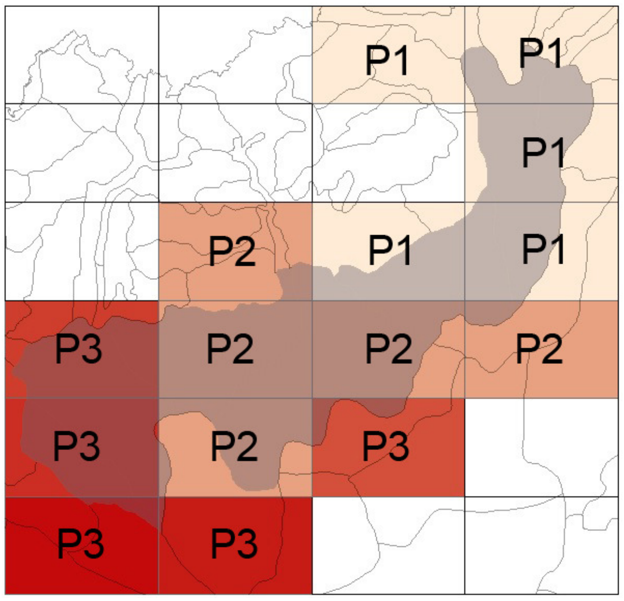

Figure 5: Distribution of the sub raster blocks to a region based processing aspect

As for data re-distribution, let's assume that, at first we have distributed data over N number of processing units, after that we have attached or removed D number of processing units into/from the distributed computing system, where $D \ll N$. In this case we have to redistribute data over the distributed computing system to keep the data distribution balanced. We may assume that data redistribution will take a lot of time, due to the fact that, we have to calculate the number of partitions and redistribute data across processing units. However redistribution of data can be considered as a maintenance service, therefore it not relevant in terms of processing algorithms.

4.4 Data associations

Until now, we have seen for every source data, there is at least one version of decomposed dataset is available on processing units. Each decomposed version is a bijective representation of the original data source, meaning that from decomposed data, we are able of reconstructing the original data looselessly. However, on algorithmic level, it is important to find the most suitable version of decomposed data for a particular processing algorithm. In order to achieve this, we need to have a versioning and tagging mechanism for these data.

Tag is non-hierarchical keyword or term assigned to version of decomposed dataset. A tag describes a property of that particular version. A tag is search-able and statistically analyzed if needed. In other word, tagging is a way of logical indexing decomposed data. Tag information on a decomposed version of an input data or output data is stored and managed by Data Catalog module. For example for one input data, the following tag and additional information will be available:

UUID	Decomposition Method	Tag
ec502529	Administrative - County	Except Budapest
87f87caf	Simple Grid based	Grid size 30km
d3a2b470	Administrative - Regional	Eastern Region

Another important aspect should be taken into consideration is the life lineage of a dataset. A processing algorithm usually uses at least two input data, resulting at least one output data. The output data should be tag with the name of the two original data and store it in a graph based database. **Neo4j** an open graph database could be used to store lineage information as well as organizing and managing decomposition information.

Figure 8 has four dataset and two processes. PS_1 takes datasets 1 and 2 as input, and after execution DS_3 dataset is created. After that, PS_2 process takes DS_3 and DS_5 dataset to create DS_4. If we would like to know, which dataset are the grand parents of DS_4, without storing dataset metadata a graph structure it could be really difficult to do so. On the other hand, by integration **tagging** functionality to dataset, we will gain an extra dimension for data search.

5. DISTRIBUTED PROCESSING FRAMEWORK

As we have already stated that, **Dispy** distributed computing framework have been chosen to run existing and newly implemented services or applications. In the following subsections some additional requirements have been added related to processing and storage topics.

5.1 Processing related requirements

Primary goal is to develop an environment where processing services or applications may run on different processing units at the same time. We are applying client/server system architecture, consisting one server unit, called master unit and client processing units.

According to user requirements for the developed solution we prioritize in certain period of the development, two main factors:

1. the processing time of an existing method compared to the distributed new approach should be faster (with constant data set).
2. accuracy of the results considered equal.

5.2 Data related requirements

Data authorization is crucial in a case of sensitive data management. Applying authentication on web based dissemination of the results and source data permit to set up the allowance of the right person to data. However in distributed file system the authentication doesn't give protection to the data storage. In future work we are going to deal with the delicate permission model allows different level of accessing data.

5.3 Proof of concept

Dispy gives us the ability to run existing services or applications on all processing units, with one single condition, that all units should be the same on software level, meaning that the number and version of all software dependencies, packages, services, executables and applications should be identical on every processing unit.

And, as a bonus, **dispy** also support Map-Reduce programming model, is anyone is missing it.

The architecture has been implemented on prototypical level which allowed us to process data sets and evaluate the performance. In the future work we plan further system development and widen processing test cases as well as its user requirements. Currently we have implemented a simple normalized difference vegetation index (NDVI) on country level dataset and made some measurements to see, if running services on distributed environment is better in performance than a very powerful server or workstation designed for GIS processing.

6. FIRST RESULTS

We have setup three client machines as processing units with 4GB of RAM and 2 CPUs, and a master unit acting as server. The server unit is reponsible for decomposing original data into smaller grids and distribute them among processing units. This decomposition demo application is a simple Python program using **GDAL** API. The NDVI calculation have been designed to send accross client processing units.

We have run NDVI processing with NIR and VIS bands as input data which have been decomposed into 22 Landsat 8 files on three processing units. Our first experiment was to detect the average runtime by running the same NDVI calculation on three processing units runs on 3×3 grids consecutively.

Node	CPUs	Jobs	SecJob	Node time sec
146.140.214.135	3	38	10.857	412.584
146.140.214.136	3	36	11.521	414.747
146.140.214.141	3	26	16.112	418.920

The average runtime is: 375.18 seconds

Our second experiement is to detect if there is any relation in decomposition grid size and runtime, therefore we have decomposed the input data into 4×4 grids resulting 11 Landsat 8 images containing NIR and VIS bands and run the NDVI calculation again:

Node	CPUs	Jobs	SecJob	Node time sec
146.140.214.135	4	58	5.753	333.675
146.140.214.136	4	57	5.718	325.932
146.140.214.141	4	50	6.565	328.249

By decomposing data into smaller grids, we have successfully gained some performance numbers, however, we should not forget on how much time did we spent on decomposing original dataset into 4×4 grids.

Our third experiment is to measure the processing time on the same dataset with commercial software ERDAS IMAGINE 2011 and 2014 on an average work station having 16GB of RAM and 4 CPUs. The first run is based on ERDAS internal data type: IMG, the second run is based on GeoTIFF. The outcome is described in the next diagram:

At last, we have made a comparision of prototype distributed system against ERDAS IMAGINE 2011. We have generated a simple model for calculation of NDVI to test our idea the distribution

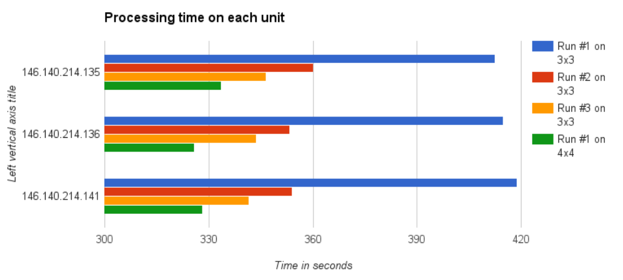

Figure 6: Processing time reduces inversely proportional as the size of grids increases

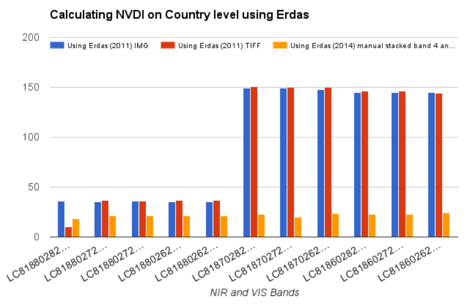

Figure 7: Processing time run by Erdas on TIFF, IMG and Stacked Bands

and process of raster data. The input data of the model were band 4 and band 5 of Landsat8 imagery without any pre-processing. The model applies virtual stacking of the imagery and after calculates the NDVI for each pixel, the output is geotiff and img. The results are compared to the developed environment processing time looks promising.

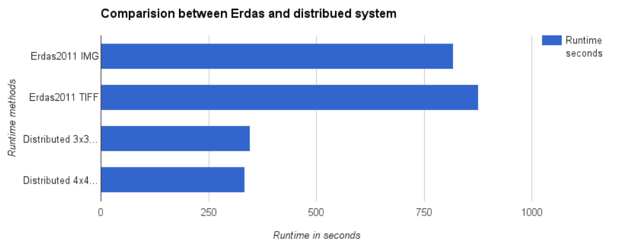

Figure 8: Comparision between Erdas and distributed system in runtime

The first results for the proof of concept distributed computing system looks promising, because we have been able to reduce the processing time to more than half comparing to commercial ERDAS IMAGINE 2011 and 2014.

ACKNOWLEDGEMENTS

This research is cofunded 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 FP7ICT20118. The research was also supported by the Hungarian Institute of Geodesy, Cartography and Remote Sensing (FÖMI), Department of Geoinformation and special thanks to Dr. Dániel Kristóf the head of the department for his directing advices.

REFERENCES

A. Dasgupta Big data: the future is in analytics, 2013.

A.Eldawy and M.F. Mokbel The Ecosystem of SpatialHadoop. The SIGSPATIAL Special, 6(3), 2014.

A.Eldawy and M. F. Mokbel SpatialHadoop: A MapReduce Framework for Spatial Data”. Proceedings of the IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April, 2015.

B. Furht Cloud computing fundamentals B. Furht, A. Escalante (Eds.) Handbook of Cloud Computing, Springer, US (2010), pp. 3–19

C.L.P. Chen and C.Y. Zhang Data-intensive applications, challenges, techniques and technologies: A survey on Big Data, Information Sciences Volume 275, 10 August 2014, pp. 314–347

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

D. Kristóf, R. Giachetta, A. Olasz, B. Nguyen Thai Big geospatial data processing and analysis developments in the IQmulus project; Proceedings of the 2014 Conference on Big Data from Space (BiDS’14) ISBN: 978-92-79-43252-1, DOI: 10.2788/1823 p. 214-217

E. Feller, L. Ramakrishnan, C. Morin Performance and energy efficiency of big data applications in cloud environments: A Hadoop case study Journal of Parallel and Distributed Computing, Volumes 79–80, May 2015, Pages 80-89

G. Wang and Q.Weng, 2014. Remote Sensing of Natural Resources. Taylor and Francis Ltd.,USA. pp 25.

Healey, R., et al. 1998. Parallel processing algorithms for GIS. Taylor and Francis Ltd., UK, pp.

J.-G. Lee and M. Kang Geospatial Big Data, Challenges and Opportunities, Big Data Research Volume 2, Issue 2 Visions on Big Data, June 2015, pp. 74–81

Kambatla et al., 2014. Trends in big data analytics. Journal of Parallel and Distributed Computing 74(7), pp 2561–2573

K. Janowicz, S. Scheider, T. Pehle, and G. Hart 2012. Geospatial Semantics and Linked Spatiotemporal Data - Past, Present, and Future, Semantic Web, Volume 3, Number 4 / 2012 IOS Press ISSN: 1570-0844

M. Hilbert and P. López The world’s technological capacity to store, communicate, and compute information Science, 332 (6025) (2011), pp. 60–65

M. R. Ghazi, D. Gangodkar Hadoop, Map/Reduce and HDFS: A Developers Perspective Procedia Computer Science, Volume 48, 2015, Pages 45-50

M. Schmachtenberg, C.Bizer, A. Jentzsch and R.Cyganiak Linking Open Data cloud diagram, 2014.

R. Piechowski The Data Life Cycle, The Art of Medicine and Technology, 2010.

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

S. Gao, L. Li, W. Li, K. Janowicz, Y. Zhang Constructing gazetteers from volunteered Big Geo-Data based on Hadoop Computers, Environment and Urban Systems, In Press, Corrected Proof, 2014.

T. White Hadoop: The Definitive Guide (third ed.)O'Reilly Media, Inc. (2012)

Z. Wu and O. Beng Chin 2014. From Big Data to Data Science: A Multi-disciplinary Perspective. Big Data Research Volume 1,Special Issue on Scalable Computing for Big Data, pp 1.