International Journal of Computer Applications (0975 – 8887)
Volume 106 – No. 16, November 2014

Parallel and Distributed GIS for Processing Geo-data: An Overview

Abdul Jhummarwala
Bhaskaracharya Institute for Space Applications and Geo-informatics (BISAG), Gandhinagar, Gujarat, India

M.B. Potdar, Ph.D.
Bhaskaracharya Institute for Space Applications and Geo-informatics (BISAG), Gandhinagar, Gujarat, India

Prashant Chauhan
Bhaskaracharya Institute for Space Applications and Geo-informatics (BISAG), Gandhinagar, Gujarat, India

ABSTRACT

Geographic Information System (GIS) is a collection of applications whose tasks include (collaborating with other systems and) gathering geographic data, store and process spatio–temporal data (geo-data) and share the derived geographic knowledge with the users and other applications. Some of the most important routine applications of GIS are spatial analysis, digital elevation model (DEM) analysis such as line of sight and slope computations, watershed and viewshed analysis, etc. GIS has became quite an important tool for geospatial sciences and has gone beyond typical tasks of mapping to performing complex spatio-temporal analysis and operations. The number of users relying upon Decision and Support Systems (DSS) built upon GIS has increased as a result of the availability of very high resolution satellite imagery and integration of spatial data and analyses with GIS packages which now satisfies the needs of many and is not just used for specialized operations. Moreover, Global Positioning Systems (GPS) in a range of mobile devices and sensors which includes updates in very short intervals has led to geo-information explosion. Parallel and Distributed Computing systems are now essential for computing over such huge amount of data and deliver faster results. The focus on development should thus be shifted from traditional GIS to Parallel and Distributed GIS as the traditional GIS systems have become quite mature and saturated while technologies such as MPI (Message Passing Interface) and GPGPUs (General Purpose Graphics Programming Units) can be readily utilized for faster geo-data processing. The performance improved using recently developed technologies such as CUDA (Nvidia GPUs), OpenCL (ATI GPUs) and Intel's Xeon Phi co-processors could be as much as ten times if not more compared to traditional Geographic Information Systems.

General Terms

Geographic information systems, Distributed GIS, Parallel GIS

Keywords

GIS, GPS, DSS, spatio–temporal data, spatio–temporal analysis, Digital Elevation Model (DEM), geospatial data, geodata, geoprocessing, OGC, Message Passing Interface (MPI), GPU, CUDA.

1. INTRODUCTION

The first widely known application of Geographic Information System (GIS) is the World-Wide Earthquake Locator. Bruce Gittings conceived its development at the School of GeoScience, University of Edinburgh in 1995. The World-Wide Earthquake Locator was meant to serve as a real-time Geographical Information System. The system was aimed at providing up-to-date information and data which were taken from the National Earthquake Information Center (NEIC) and also providing the detailed dynamic maps of earthquakes across the world. The same were also published over the World Wide Web (WWW) [1].

The first generation of GIS which can also be termed as Desktop GIS has gone far beyond evolving into Enterprise GIS or Corporate GIS to Web GIS. The system architecture of Corporate and Enterprise GIS put all the data and its processing on the centralized server. This client-server approach resolved the issues of data consistency and data sharing which had emerged due to increase in volume and complexity of data. Collaborating and coordinating data between different entities of an Enterprise GIS was then possible for generation of information and in turn better planning and supporting decision making. Historically, the GIS used to be just a support system for the tasks of Remote Sensing data analyses but since the inception of OGC (Open Geospatial Consortium; previously known as Open GIS Consortium), various service architecture and frameworks, spatial databases and data formats have been standardized. This has resulted into many OSGeo (Open Source for Geospatial) packages not just supplementing Desktop and Enterprise GIS but has also led to the evolution of Web GIS. The standardization was necessary due to the distribution of data and autonomous services provided by various governmental and non-governmental organizations or groups over the web. The standards were aimed at making such services and data interoperable. Recently, spatial support has been integrated into commercial database managements systems such as Oracle and Microsoft SQL Server [4]. The spatial database system may have the integrated spatial data support or may be added by use of extensions such as in the case of IBM DB2 and Open Source PostgreSQL DBMS which use PostGIS addon [3]. Various ISO/IEC standards formalize the storage and support for different two dimensional geographic data which includes point, line, polygon, etc. OpenGIS®'s Simple Features Implementation Specifications is one of the ISO/IEC standards which defines interfaces that enable transparent access to geographic data held in heterogeneous processing systems on distributed computing platforms [2]. Most of the GIS software and databases that support the Simple Feature Specification standard provide application programming interfaces (APIs) provide for publishing, storage, access and simple operations on GIS features. This allows interoperability and collaboration between different GIS software, applications and database management systems built across different platforms, of course using different technologies.

Parallel, Distributed and Cloud Computing offers resolution to many limitations possessed by commodity hardware. Each of these computing techniques has its own set of strengths due to their architectural differences. These set of strength not only provides resolution to limitation of storage and processing of conventional data but also provide scope to modify them for supporting spatial querying cases or spatial operations. Due to the large size of spatial datasets and computationally intensive nature of spatial operations, it is now indeed necessary to reap the benefits of parallel and distributed computing for spatial analysis.

2. OVERVIEW

Geographic information systems are in the midst of information revolution. Geographic data also known as spatial/geo-data (i.e. data which has an associated location component) is now being collected by various positioning technologies and devices such as mobile devices (GPS), sensor networks, RFIDs, etc in addition to the specialized remote sensing capabilities of satellites. This has lead to an unprecedented increase in the amount of geo-tagged data (images, web-posts, tweets, etc) whose use is not just restricted to the field of remote sensing (terrain, weather and climate data, etc) but also being utilized by other scientific applications such as those related to Medical Imaging, Marketing, Geographic Intelligence, etc.

The Earth Observatory System (EOS) satellites of NASA collect roughly about 1 TB of spatial data every day. Overall the data collected and stored by Earth Observing System's Data and Information System (EOSDIS) amounts to more than 3 petabytes [5]. Geographically Distributed systems have been developed for storing such vast amount of data, also known as BigData [23]. The branch of Digital Medical Imaging today generates extremely high resolution images of trillion pixels and hundreds of thousands of such images are taken regularly. The digital slide images taken pathology instruments are very large, with dimensions that routinely exceed 100,000 x 100,000 pixels [6]. Apart from the collaborative efforts such as OpenStreetMap, techniques such as montage, mosaicing and blending also generate very large images from smaller images (tiles) to create a large and complete view. Thus due to such an increase in volume of spatial data, techniques used for handling, visualizing, managing and processing terabytes to petabytes of data cannot be ignored. This has resulted into two major challenges, the same like every other high performance application has, storage[7] and processing[8] of spatial data.

A decade earlier, MapReduce [9], a programming model and an associated implementation for processing and generating large data sets have been designed by engineers at Google to keep pace with the exponential growth in data to be stored, indexed and processed, etc collected by their web-crawlers. At the same time another framework for storage and processing of large scale data sets over commodity hardware named Hadoop was in development for search engine at Yahoo! The source code of Hadoop was made Open in 2009 and since then Apache Software Foundation has been developing and supporting Apache Hadoop as one of its top level project[10]. Apache Hadoop consists of Hadoop Distributed FileSystem (HDFS) and an implementation of MapReduce model. It has become an enterprise-ready cloud computing technology. It is becoming the industry de facto framework for big data processing [9].

There have been significant developments in storing and handling of BigData. Declarative query interfaces such as Apache Hive, Apache Pig, and SCOPE (Structured Computations Optimized for Parallel Execution) have brought the large scale data analysis (i.e. data from access logs, web analytics, etc) one step closer to common users by providing high level, developer friendly programming abstractions (such as Pig for SQL on Hadoop) to MapReduce [12]. The interesting fact here to know about is that these frameworks have been specifically designed to compute upon textual data (from streams such as output of webcrawlers, etc).

Spatial data analysis is the application of techniques to data (geographic) over its topological, geometric or geographic component or dimension. Spatial Analysis includes a variety of operations such as spatial characterization, spatial dependency/correlation, etc. Spatial Analysis can be performed using the spatial constructs and spatial querying engine provided by the underlying framework. Spatial and non-spatial operations can be divided into following five major categories [11] [19]:

1. Feature aggregation queries (non-spatial queries), for e.g. queries for finding mean values of attributes or distribution of attributes
2. Fundamental spatial queries, including point based queries, containment queries and spatial joins including union, intersection, difference and XOR
3. Complex spatial queries, including spatial cross matching or overlay (large scale spatial join) and nearest neighbor queries
4. Integrated spatial and feature queries, for e.g. feature aggregation queries in a selected spatial region, and
5. Global spatial pattern queries, for e.g. queries on finding high density regions, or queries to find directional patterns of spatial objects.

As per our previous discussion spatial constructs have been provided in the recent versions of Oracle, PostGIS and SQL Server database management systems. These constructs have been extensively used and are still under development for spatial support. Computations performed on Spatial datasets using a single machine has its own inherent limitations arising out of memory and disk space. Moreover, the spatial framework used might not be able to support the size of the dataset. For example, a spatial query which performs a spatial join between two datasets having tens of thousands of attributes might take exponentially long time or might require more than maximum available memory and disk space supported by the hardware.

In this paper we discuss the relevance of various distributed and parallel technologies and frameworks that have extended the traditional desktop based GIS systems to scale with the increase in amount of data coming from millions of handheld portable, autonomous devices, sensor webs and very large scale imagery generated from high precision devices and remote sensing satellites. These precision devices and the latest advances in telescope design and satellite imagery have dramatically improved the accuracy and spatio-temporal scope of the data. Exabytes of data is accessible through internet in the public domain and most of it can be characterized as geo-data (geospatial data) as every element of data has a geographic component such as an address, an area reference or a map coordinate. Whether it is massive remotely sensed images, billions of records generated everyday by adclick and tracking services or data collected through/for location based services, it is used for better facilitation for the users. The scientific use of this data is predicting earthquakes, tsunamis, environmental issues, etc. The commercial and domestic use of the data includes the trend analysis for various business entities. The same is again useful to governments for urban planning, transport, agriculture, criminology, etc all for making human lives more convenient.

Several frameworks have been specifically designed to tackle the shortcomings of the commodity hardware while dealing with such massive amount of data and are not fault tolerant, fails to scale with the increase in demand, accessibility and interoperability. The most prominent of these models are built upon the strength of high performance technologies which include cloud computing, hadoop based map-reduce and graphics processor based CUDA and OpenCL environments.

The rest of the paper is organized as follows:

Section 3 provides background of GIS raster and vector data.

Section 4 and 5 reviews the literature briefly for high-performance architectures developed for specific use cases (MapReduce/Hadoop) and related work.

Section 6 discusses the shortcomings of the approaches and future research directions.

3. BACKGROUND OF RASTER AND VECTOR DATA

Real world digitized geo-spatial data can be stored in two basic forms with an optional temporal component viz. raster and vector.

3.1 Raster Data

ESRI (Environmental Systems Research Institute) defines Raster as a spatial data model that defines space as an 2D array of equally sized cells arranged in rows and columns, and composed of single or multiple bands. It describes the information through values stored in pixels. The spatial resolution of a raster image is dependent upon the resolution of the acquisition device such as Optical Sensor, CCD Device or other imaging device and its quality upon the source of data. Raster data requires more storage space than vector data. Each pixel that represents its attribute value is known as a cell. A group of cells belong to a grid in the raster matrix. The cells in a grid have similar values to represent the same type of geographic feature. The co-ordinates of the cells are in the positional ordering of the matrix. The most widely used Raster file formats are GeoTiff (an extension of TIFF format to accommodate GIS Metadata), JPEG2000, formats standardized by National Geospatial Intelligence Agency (Compressed/ARC Raster Graphics) and proprietary file formats such as those supported by products of ESRI and Hexagon Geospatial, etc. The specific examples of such raster images known to us are cadastral, aerial imagery, digital elevation models, etc. Images are captured at various scales, resolutions and are archived at different locations/formats. Heterogeneity problems such as different formats, encoding schemes, temporal characteristics, etc arise frequently due to such non-structured data. OGC introduced GML (Geography Markup Language), WKT (Well Known Text) and WKB (Well Known Binary) as interoperable alternatives to various types of non-structured data formats. Studies have been conducted such as "Retrieving and indexing spatial data in the cloud computing environment" [15] and "Demonstration of Hadoop-GIS: A spatial data warehousing system over mapreduce" [11] which aim at distributed storage as well as processing of extremely large datasets.

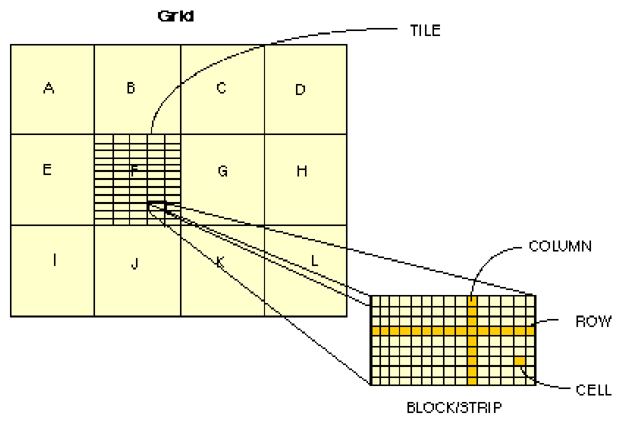
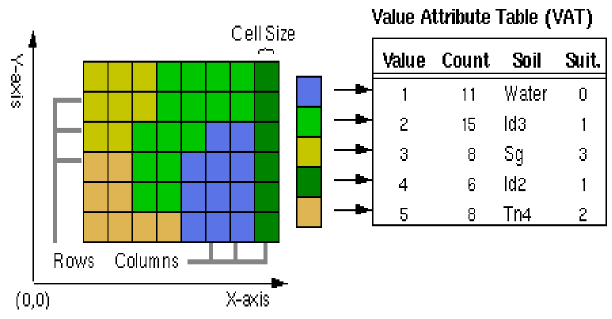


Fig 1: Representation of Cells in a Raster Grid [14]



Value	Count	Soil	Suit.
1	11	Water	0
2	15	Id3	1
3	8	Sg	3
4	6	Id2	1
5	8	Tn4	2

Fig 2: Value Attribute Table showing unique records for Grid Cells

3.2 Vector Data

The vector data describes information through geometric shapes such as point, line, multiline, polygons and other complex shapes. It is mostly prepared through surveying and digitization of maps manually or through supervised/non supervised automated programs. Pattern Recognition and Image Processing techniques are used to convert raster formats into vector formats whereby vector features such as lines and polylines are identified by the tracing program. Recent advances in image processing algorithms helps to convert much of the raster data into vector formats with very high acceptable accuracy [18].

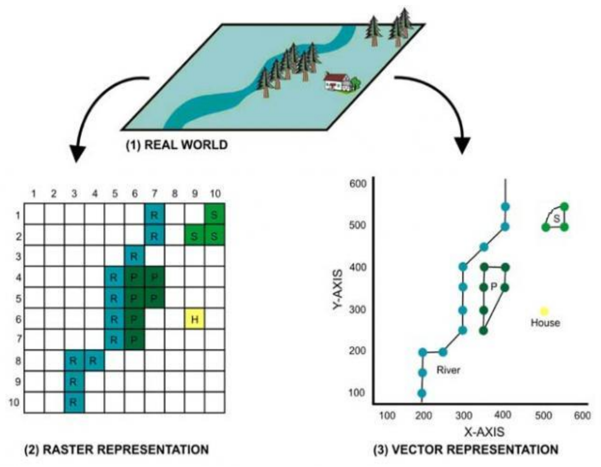


Fig 3: Information from Map represented in Raster and Vector formats

Vector formats explicitly stores the coordinates of the geometry or the geometries may be defined using mathematical formulas unlike a Raster format. Vector data enables the GIS or CAD (Computer Aided Design) systems utilizing it to be more flexible in terms of resolution, scaling and conversions. Moreover it is very easy to embed metadata (data about data) in vector file formats. Some examples of metadata includes map legend, representation of different map elements, publishing date, projection and coordinate system, etc[13]. Most widely used and standardized vector data formats are GML (Geography Markup Language), ESRI's shapefile format, etc. Figure below shows representation of map information in Raster and Vector formats.

Most of the existing data formats (raster and vector) can be processed through a variety of processing techniques[16] specifically designed for handling such data and image formats. These may include preprocessing, digital enhancements, geometric corrections (via Ground Control Points), Map Projections, etc. Massive amount of data is accumulated everyday due to the advancements in the sensor technologies not just limited to imagery (raster data in form of millions of photographs and images shared everyday on the internet.) but also spans other domains such as those using temperature, pressure, light, sound and other environmental aspects such as humidity, aerosols, etc. Much of this data sought from wireless sensor networks is spatially linked to provide better judgment in manufacturing and machinery applications.

4. HIGH PERFORMANCE GIS (LITERATURE REVIEW)

The inception of Hadoop framework took place at Yahoo! in 2005 to support distributed computation of the data gathered by the web-crawlers. The architecture of Hadoop was similar to the many open-source and closed source implementation of Map Reduce frameworks and also known as MapReduce, a Google technology but the term has since been generalized for distributed applications due to the massive and unprecedented wide-spread acceptance and usage of Hadoop. This was the result of the move of Yahoo! to make the source code of Hadoop Open-Source and available to general public in 2009.

Hadoop is a distributed Computing platform written in JAVA and provides a Compute and Storage architecture similar to those being used by Google, Yahoo and others for BigData processing.

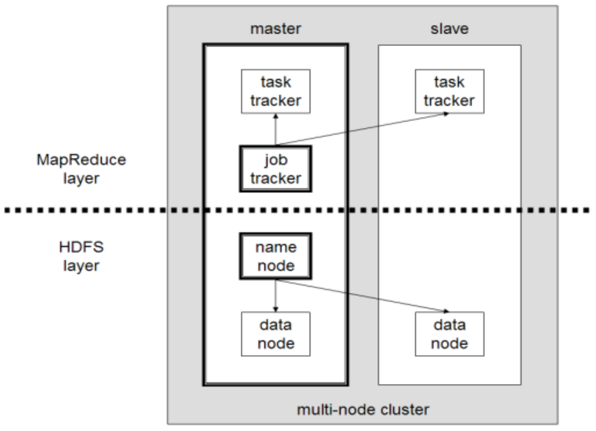


Fig 4: A multi-node Hadoop cluster having multiple data-nodes

The Hadoop Distributed File System (HDFS) and MapReduce model forms the most important entities of the architecture. Hadoop extensively makes use of Linux environment for running shell scripts and depends on remote connectivity services only available in Linux but can be run on Windows environment using CygWin [20].

4.1 MapReduce

Map reduce is a technology that was first widely publicized by Google. The Map reduce terminology is similar to divide-and-conquer methods most prominently used by the parallel and distributed architectures. The MapReduce programming model relieves the programmers from the underlying issues of parallel and distributed architectures and allowing them to develop the application. As the term suggests, the model is based upon two most important phases, the Map and the Reduce.

The Map phase splits the input data and the processing (user submitted jobs) to be done into tasks which are then assigned to Worker Nodes/Task Trackers for computation. Hadoop takes care of localizing the Data so that the processing is sent to a Worker node that has the Data and reduces the communication overhead. Each worker node also acts as a Data node. The Task Tracker heartbeat (a service monitoring the execution, status of task and Worker Node being online) is continuously monitored by the Job Trackers, the master node in the Hadoop Environment. The Job Tracker is also responsible for admission control, tracking the liveness of Task Trackers, reporting job status to users, etc. [10]

Fig. 5 represents a MapReduce model for performing some computation upon Text input. Textual input is the simplest of all input forms to MapReduce as the data from it can be easily divided (to store on multiple data nodes) and the intermediate results merged easily for the required output. We would like to highlight both the vertical and horizontal partitioning of data (slices) which is simple to implement on Text.

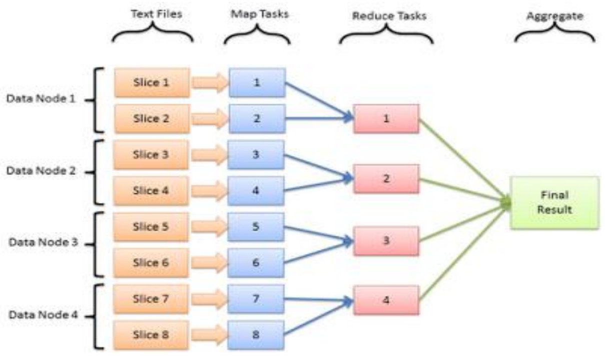


Fig 5: Simple MapReduce Operations upon Textual input

4.2 YARN (Yet Another Resource Negotiator) and its Applications

With the introduction of the new architecture of Hadoop's approach to MapReduce, also known as YARN[21], it is now possible to deploy other parallel and distributed programming frameworks such as Dryad, Giraph, Hoya, Reef, Spark, Storm and Tez over Hadoop. The new architecture separates the monolithic programming model from the resource management infrastructure. The scheduling function rather than being system wide is now on a job basis as multiple applications require its own scheduling while running over YARN. Applications following the MapReduce paradigm can also be deployed easily over YARN. YARN promises to have full backward compatibility with other existing applications and legacy Hadoop implementations.

How did YARN come into existence? Applications deployed over legacy Hadoop used to create clusters (using a group of nodes) and transfer the data to HDFS (Hadoop Distributed File System), perform map-reduce operations and release the cluster/nodes. This scenario resulted in Hadoop and HDFS to being extended to support a multi-tenant model whereby shared clusters were formed to deploy different applications. The sharing of resources lead to creation of resource pools [23] and features such as permissions, quotas per user and per job, etc. were added to Hadoop. It was also possible to deploy applications using different versions of Hadoop.

Yahoo extended the legacy MapReduce Hadoop and developed Hadoop on Demand (HoD) using schedulers such as Torque and Maui to support this dynamic resource allocation. HoD still lacked the isolation and security requirements due to the federated (tightly coupled) architecture of Hadoop Installations. Based upon Torque, HoD lacked allocation of nodes taking into consideration, the locality of Data, which formed the strength of Hadoop. Hadoop was built around the philosophy of taking computation to data (and in turn minimize the communication cost). It was also not possible to resize the clusters during the execution of fewer Reduce operations. While there are applications that require hundreds and thousands of nodes during peak calculations (processing), the same may require a handful of nodes for the very few Reduce operations e.g. Application workloads that can be represented in the form of DAGs (Directed Acyclic Graphs) having a high degree of Fan-In or Fan-Out. Due to the static allocation of nodes(resources), if the workflow possesses map or reduce operation as its bottleneck, it would lead to poor cluster utilization and wastage of useful computation power.

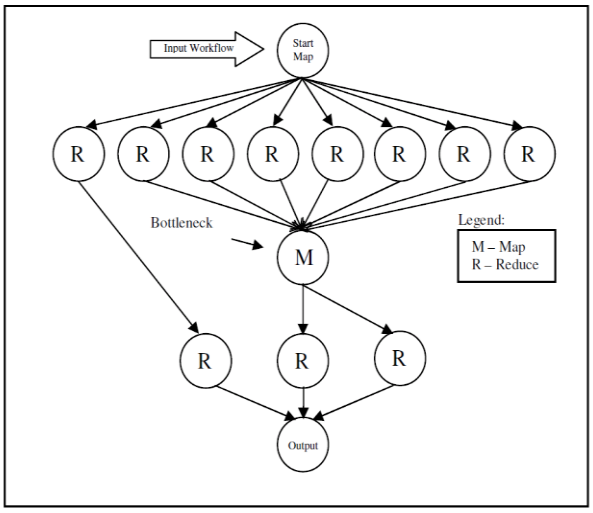


Fig 6: DAG Representation of MapReduce Operations

HDFS evolved gradually due to the increase in storage requirements of the applications which accumulate terabytes of data everyday resulting in the data-store of petabytes while the Job Tracker was not much modified and still remained primitive. Application specific customizations to the JobTrackers lead to bugs and failures in workflow executions. Moreover as the JobTracker, a single point of failure was shared across applications inside a shared cluster, there was too much dependency and overload of large number of Job Monitoring lead to unnecessary delay in allocation of resources. A strong and scalable authentication and authorization model was required for securing applications in the multitenant clusters. Legacy Hadoop lacked support for different programming models so users would write "MapReduce" equivalents for jobs required for Machine Learning and Graph Algorithms, all leading to poor resource utilization. The following table summarizes features supported by the major releases of Hadoop.

Table 1. Classic MapReduce vs. YARN

Feature	Classic MapReduce (Hadoop 1)	YARN (Hadoop 2)
Authentication for HDFS	No	Yes
HDFS federation	No	Yes
HDFS high-availability	No	Yes
Separate Application Manager and Resource Manager	Scheduler takes care of applications and resources (and is the bottleneck for scalability to >1000s of nodes)	Yes
Task Size (w.r.t running time and I/O)	Large	Large and Small
Supported Applications	Single (MapReduce)	Multiple Paradigms
Backward Compatible	-	Yes (Supports Classis MapReduce as one of the paradigms)

4.3 HDFS (Hadoop Distributed File System)

Hadoop uses HDFS as its primary distributed storage system [22]. HDFS can function as a general purpose distributed storage filesystem or conjunctively used with Hadoop Clusters on commodity hardware. It is highly similar to existing distributed file systems but is highly fault tolerant and provides streaming access to data. HDFS is designed for scalability and supports multi-gigabyte to multi-terabyte files with emphasis on low latency of data access. HDFS is based upon the simple coherency model of write-once-read-many (WORM) which enables high throughput but at the same time restricts the updations/appends which are usually sluggish. HDFS has been designed on the foundations of data locality and thus it serves best for applications not requiring frequent updates (writes).

HDFS is designed to be a distributed file system and handle extremely large files that are not supported by normal file systems. These files are divided into typical chunk/block size of 64MB and stored in datanodes. A Cluster of Data Nodes forms the HDFS cluster which is able to handle files of size in the range of gigabytes to even terabytes. HDFS achieves fault tolerance by replicating (default value: 3) blocks/chunks across datanodes dynamically. There exists a single NameNode in the Hadoop Infrastructure and is the main metadata server. As the namenode is the sole entity storing and managing the file system metadata, it becomes a huge bottleneck in supporting a large number of small files. We do acknowledge that the bottom up design of HDFS has been to store a few huge files rather than millions and billions of tiny files. The failure of the NameNode results in the failure of the application. A secondary namenode can also present in the environment to cope up with the failure of the primary NameNode which clones the directory structure of the primary NameNode and serves as a backup for restarting the failed primary NameNode without having to rebuild the HDFS. This has been tackled in the Hadoop 2.0 through use of separate NameNode spread across multiple namespaces.

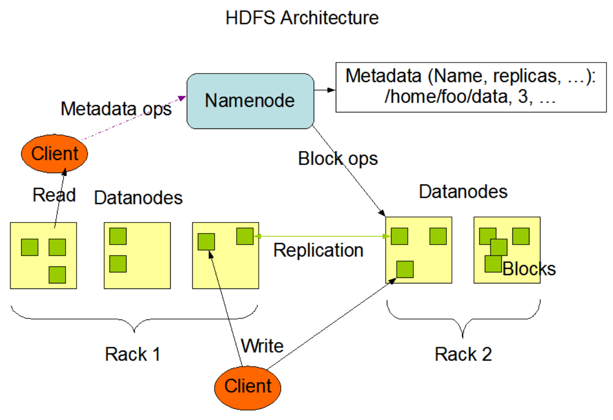


Fig 7: HDFS Architecture (http://hadoop.apache.org)

Classes within *DFSInputStream* and *DFSOutputStream* namespaces handle the retrieval and storage of files in the HDFS in form of blocks/chunks. The data of file from the client requesting a File Write is cached by *DFSOutputStream* to the local filesystem. After the data write operation reaches the block size of HDFS, *DFSOutputStream* requests the NameNode to add the block to the file on HDFS. NameNode returns a pipeline of DataNodes to whom the client writes the block. After successful writing and replication, the DataNode sends a *blockReceived* message to the NameNode and an acknowledgement to *DFSOutputStream*.

The replication of the block as specified by the HDFS user is done by the other DataNodes in the pipeline until the specified replicas of the block have been written. Reading of a file from HDFS requires the Client to request the NameNode for the desired block. The NameNode returns a list of locations of the replication sites. The client can then fetch the block from one of the replicas. Corrupt block received has to be reported by using reportBadBlocks to the NameNode.

As Hadoop is built upon the philosophy of bringing Computation to Data, it minimizes the read data from the network. Depending upon the scalability required for applications, HDFS instances run on nodes that may spread across many racks leading to IO between different racks. In most cases this IO is slower than communication between nodes in the same rack [22]. HDFS provides APIs to customize applications that can utilize the rack id's for better placement of data. This does improve the overall efficiency of the infrastructure but is quite unnecessary for high speed networks such as 10Gigabit Ethernet, Infiniband and their successors.

The performance analysis of the HDFS over different types of networks and its optimization is quite viable to devise distributed file systems specifically for today's high speed networks and support millions of files (small and large) which are simultaneously used by many interoperating applications.

4.4 Apache Oozie

Apache Hadoop is complimented by Oozie, a server based Workflow Engine that also acts as a Coordinator Engine and a Bundle Engine [10]. Oozie has been specially designed for running workflow jobs executing as Hadoop's Map/Reduce or Pig jobs. Oozie workflow applications contain files such as .xml, .jar, .so, etc and are installed in the HDFS itself. The workflow jobs are specified in hPDL (a XML Process Definition Language). The workflow jobs can be executed on triggers such as time and data availability, if not manually. This is necessary to coordinate between jobs that execute in continuity with other Map/Reduce operations across the depth in a DAG representation.

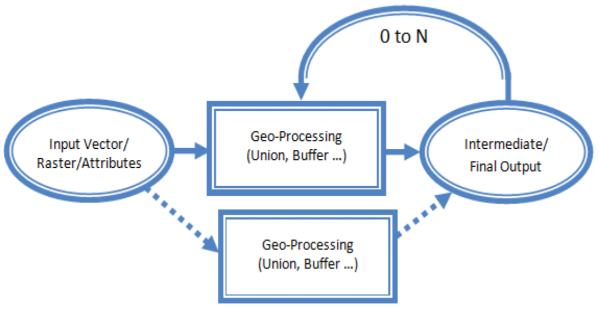

0 to N
Input Vector/ Raster/Attributes
Geo-Processing (Union, Buffer ...)
Intermediate/ Final Output
Geo-Processing (Union, Buffer ...)

Fig 8a: A Simple GIS workflow model representation

The latest version of Oozie also allows coordination between different applications. The users can control the flow of the executing workflow jobs as Oozie provides better operational control for managing jobs such as start/stop/suspend/resume and restart, if required. There are numerous GIS applications that can advantage from this capability. Most of the GIS applications require more than one geoprocessing function (Intersect, Union, Buffer, Clip, etc) to be applied to the input in a sequence (or a cycle) for performing complex Geo-processing known as a workflow. Such Workflow applications (containing cycles) can always be reduced to the form of a DAG. The DAG then may contain either Hadoop Jobs, or Pig Jobs or Streaming applications or sub-DAGs. Oozie lacks a visual workflow model builder and interoperability with OGC standards. Fig. 8a represents a sample GIS workflow.

It is indeed important to highlight here that cycles in workflows are not supported which makes us to redesign applications that require recursive executions of jobs. Oozie maintains the states (PREP, RUNNING, SUSPENDED, SUCCEEDED, KILLED and FAILED) for workflow jobs.

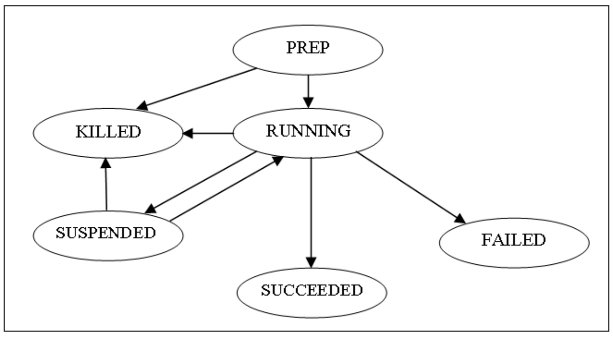


Fig 8b: Oozie workflow job state valid transitions and lifecycle

The DAG representation in Fig 6 shows a Fork and Join example for an Oozie workflow. It has '1' Starting Job (Start Map) generated from the Input Workflow, '8' Jobs (R) that are executing in Parallel which are created by 'Start Map', '1' Reduce and Map Job (M) that collects results of '7' jobs (R) while '1' Job (R) is still running. After the completion of the '1' (R) Job that was still running and the '1' Reduce and Map Job (M), '3' Reduce Jobs (R) again start in parallel. The last Job (Output) finally collects the results and provides the output.

4.5 Motivations

We have already discussed what makes GIS a perfect usecase for the parallel and distributed computing architecture. Spatial operations involve analysis over two categories of data; vector (points, lines and polygons, etc) and raster (image data). Let us now discuss more regarding the approaches and directions that have been implemented for various use cases which includes spatial operations such as polygonal overlays, spatial range queries, spatial joins such as Intersection, Spatial Cross Matching, global spatial pattern discovery, etc.

As previously discussed, high performance architecture is required for querying on large volumes of spatial data which is important in many scientific as well as commercial domains. This presents with two major challenges; the storage and management of spatial data and high computational requirements of spatial queries. Parallel RDBMS (Relational Database Management System) and SDBMS (Spatial Database Management System) exists to achieve scalability and reduce the I/O bottleneck by distribution and aggregation of data across multiple systems but are not optimized for performing computation of spatial queries.

5. RECENT ADVANCES IN HIGH PERFORMANCE GIS

GeoJinni aka SpatialHadoop[17] has been designed upon Apache Hadoop to perform batch analysis of large spatial data. GeoJinni alike other Hadoop customizations and unlike PostGIS is not designed and is not capable of running interactive queries upon small datasets. The spatial indexes generated (Grid File, R-tree and R+-tree) and stored in HDFS are accessible to applications using GeoJinni API and it also provides interfaces with other Hadoop tools and related projects such as Apache Hive, Pig or Hbase.

Pigeon [18] extends Pig by including two additional components viz. spatial datatypes and spatial functions. Pigeon supports the standard OGC datatypes such as Point, Linestring, MultiLinestring, Polygon, MultiPolygon, and GeometryCollection in addition it can import spatial objects stored in the Well-Known Text (WKT) and Well-Known Binary (WKB) formats. Pigeon relies upon the ESRI Geometry API 1.0 to support spatial objects. Pig has been extended through its support of User Defined Functions (UDFs) which can be used to extend the existing operators such as filter, join and group by. There are four groups of spatial functions implemented in Pigeon: (1) Basic spatial methods such as Eval (Area occupied by a spatial object), (2) Spatial predicates such as those defining relationships among spatial objects such as (Do A intersects B), (3) Spatial analysis which operates upon 1+ spatial objects, and (4) Aggregate functions that computes upon a set of spatial objects (e.g. Convex Hull). User can create customized Pigeon scripts with the same syntax as that of Pig to perform simple spatial operations involving a single MapReduce job while complex spatial operations can be performed using multiple MapReduce Jobs.

Hadoop-GIS [11] is a scalable and high performance spatial data warehousing system for running large scale spatial queries utilizing the distributed storage and MapReduce capabilities of Hadoop. Hadoop-GIS support spatial data types, spatial operators and functions. Spatial partitioning is achieved through global partition indexing and customizable on demand local spatial indexing. It integrates a customizable spatial query engine **RESQUE** (**Re**al-time **S**patial **Qu**ery **E**ngine) similar to Pigeon but in addition supports data-compression resulting into low I/O overhead and communication. RESQUE supports implicit parallel spatial query execution on MapReduce through use of a declarative spatial query language upon HiveSP (Spatial Hive). Hive Query Engine is also extended to support spatial queries. In addition to supporting complex spatial partitioning and querying, major operators and functions from ISO SQL/MM are also supported by the spatial querying language (QLSP) used with HiveSP. This is achieved by extending HiveQL with spatial constructs, spatial query translation and execution. Hadoop-GIS employs distributed spatial queries through spatial workflows include joins, containments while it also effectively amends query results efficiently handling boundary objects.

6. CONCLUSION AND FUTURE WORK

Billions of devices are generating unstructured data every day. Most of this data is utilized by application for provision of user friendly services personalized according to their location. Terabytes to petabytes of data is required to be processed in seconds to obtain time-relevant smart output. Existing infrastructures have to adapt to such requirements as users are unaware about the complexities of the geoprocessing tasks. Methodologies adopted for BigData have kept up with the increasing data but the scope for complexities associated with the real-time analysis of such data resulting in quality information is still open.

In this paper we reviewed quantity of parallel and distributed tools and technologies which can be adapted for processing of Geo-data. Several of them are based on Hadoop, which works on principle of MapReduce (Divide and Conquer). Although, MapReduce is more than a decade old, the paradigm has not evolved for use with GIS applications. Recent developments in variety of distributed technologies (mostly by Apache such as Hive, Pig, etc) can be modified and utilized for creation of Spatial Data Infrastructures and GIS applications. In addition, modern HPC (High Performance Computing) systems are also designed around CUDA (from Nvidia) and OpenCL architectures for faster processing. Several experimentations have already been conducted for GIS with CUDA but are still not available for general use. Further research needs to be carried out for porting processing of GIS applications to GPGPUs.

The utilization of both the parallel and distributed architectures remain open for geo-processing owing to the strengths of both. The utilization of parallel architectures for compute intensive geoprocessing tasks and distributed processing for data intensive tasks. Further of our work would be based upon dynamic visual workflow based geoprocessing engine developed over Hadoop and CUDA concurrently for executing workflow processes.

7. ACKNOWLEDGMENTS

We would like to take an opportunity to express our gratitude and deep regards to Bhaskaracharya Institute for Space Applications and Geo-informatics (BISAG), Gandhinagar, India for providing the adequate resources and work environment. We would also like to thank Mr. T. P. Singh, Director, BISAG for his invaluable support. We would additionally duly recognize the feedback and constant encouragement provided by Mr. Miren Karamta and Mr. Punit Lalwani, Project Scientists at BISAG, Gandhinagar. Their perceptive criticism and innumerable support kept us working and concretizing this research work.

8. REFERENCES

[1] The World-Wide Earthquake Locator. http://tsunami.geo.ed.ac.uk/local-bin/quakes/mapscript/home.pl

[2] Simple Features for OLE/COM. http://www.opengeospatial.org/standards/sfo

[3] PostGIS - Spatial and Geographic Objects for PostgreSQL. http://postgis.net/

[4] Li Yingcheng, Li Ling. 2012 Research On Spatial Database Design and Tuning Based on Oracle and ARCSDE, International Society for Photogrammetry and Remote Sensing, Volume XXXV Part B4, 2004. XXth ISPRS Congress Technical Commission IV.

[5] The Open Data project. http://data.nasa.gov/about/

[6] A. Aji, F. Wang, and J. H. Saltz. 2012 Towards Building A High Performance Spatial Query System for Large Scale Medical Imaging Data. In SIGSPATIAL/GIS pages 309–318.

[7] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang, and J. Saltz. 2013 Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce, Proceedings of the VLDB Endowment Vol 6.

[8] Jeffrey Dean, Sanjay Ghemawat 2008 MapReduce: Simplified Data Processing on Large Clusters, Communications of the ACM, Volume 51, Issue 1.

[9] Hadoop at Yahoo!. https://developer.yahoo.com/hadoop/

[10] The Apache Software Foundation. http://apache.org/

[11] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang, and J. Saltz. 2013 Demonstration of Hadoop-GIS: A Spatial Data Warehousing System Over MapReduce, Proceedings VLDB Endowment.

[12] Jingren Zhou, Nicolas Bruno, Ming-Chuan Wu, Per-Ake Larson 2012 SCOPE: Parallel databases meet MapReduce, The VLDB Journal, Vol 21, Issue 5.

[13] CLUP GIS Guidebook: A Guide to Comprehensive Land Use Data Management http://www.cookbook.hlurb.gov.ph/book/

[14] Esri Grid format http://resources.arcgis.com/en/help/main/10.1/index.html

[15] Yonggang Wang, Sheng Wang, Daliang Zhou. 2009 Retrieving and Indexing Spatial Data in the Cloud Computing Environment, Lecture Notes in Computer Science Volume 5931, 322-331.

[16] Michael Paul Deskevich, Method and system for processing raster scan images. http://www.google.com/patents/US8731309

[17] Ahmed Eldawy, Mohamed F. Mokbel. 2013 A Demonstration of SpatialHadoop: An Efficient MapReduce Framework for Spatial Data, Proceedings of the VLDB Endowment, Vol 6 Issue 12.

[18] Ahmed Eldawy and Mohamed Mokbel. 2014 Pigeon: A Spatial MapReduce Language, In Proceedings of the IEEE International Conference on Data Engineering.

[19] Ahmed Eldawy, Yuan Li, Mohamed F. Mokbel. 2013 CG_Hadoop: Computational Geometry in MapReduce, Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems

[20] Hadoop 2 On Windows https://wiki.apache.org/hadoop/Hadoop2OnWindows

[21] Vavilapalli V.K., Murthy A.C., Douglas C., Agarwal S., Konar M. 2013 Apache Hadoop YARN: Yet Another Resource Negotiator, Proceedings of the 4th annual Symposium on Cloud Computing.

[22] The Hadoop Distributed File System: Architecture and Design https://svn.eu.apache.org/repos/asf/hadoop/common/tags/release-0.16.3/docs/hdfs_design.pdf

[23] Fay Chang, Jeffrey Dean, Sanjay Ghemawat et al. 2008 Bigtable: A Distributed Storage System for Structured Data, ACM Transactions on Computer Systems (TOCS), Volume 26 Issue 2