HiGIS: An Open Framework for High Performance Geographic Information System

Wei XIONG*, Luo CHEN
College of Electronic Science and Engineering, National University of Defense Technology, 410073, Changsha, China
{xiongwei, luochen}@nudt.edu.cn

***Abstract*—Big data era expose many challenges to geospatial data management, geocomputation and cartography. There is no exception in geographic information systems (GIS) community. Technologies and facilities of high performance computing (HPC) become more and more feasible to researchers, while mobile computing, ubiquitous computing, and cloud computing are emerging. But traditional GIS need to be improved to take advantages of all these evolutions. We proposed and implemented a GIS married with high performance computing, which is called HiGIS. The goal of HiGIS is to promote the performance of geocomputation by leveraging the power of HPC, and to build an open framework for geospatial data storing, processing, displaying and sharing. In this paper the architecture, data model and modules of the HiGIS system are introduced. A geocomputation scheduling engine based on communicating sequential process was designed to exploit spatial analysis and processing. Parallel I/O strategy using file view was proposed to improve the performance of geospatial raster data access. In order to support web-based online mapping, an interactive cartographic script was provided to represent a map. A demostration of locating house was used to manifest the characteristics of HiGIS. Parallel and concurrency performance experiments show the feasibility of this system.**

***Index Terms*—high performance computing, geographic information system, geocomputation, communicating sequential process.**

I. INTRODUCTION

It has become common sense that the geospatial data is fairly "big" in terms of volume, increasing velocity and variety. Therefore, analyses, simulation, mining tasks on geospatial data sets captured with ubiquitous location-aware sensors require massive computing power, storage space and communicating bandwidth. High Performance Computing (HPC) cluster with large amount of CPUs, memory, hard disks and high-speed networks provides a suitable parallel processing environment to solve complex geospatial data processing problems. In traditional desktop-based Geographic Information System (GIS), the processing paradigm of a complex geospatial procedure is in sequence on account of the personal computer hardware architecture. However, such paradigm is too low-efficiency to be applied on HPC.

Traditional GIS e.g. ESRI ArcGIS, Quantum GIS running as a standalone desktop program takes good advantage of the computing power of personal computers. The performance of such kind of GIS with regards to the geospatial data throughput, visualization as well as spatial analysis is restricted by the capability (number of cores, CPU frequency, memory speed, Input/Output latency and bandwidth) of the local computer. Although there are some server-based systems e.g. ArcGIS Server, Geoserver, Mapserver etc, can provide web services conformed to Open Geospatial Consortium (OGC) standards, which support interoperable solutions that "geo-enable" the Web, wireless and location-based services and mainstream IT (http://www.opengeospatial.org/). These systems put their main focus on providing mapping (WMS) or geospatial data service (WFS and WCS) via HTTP. However, to do a complicated geospatial computing mission is still a tough nut to crack. In recent years, some online GIS-like system emerged as the rapid rise of cloud computing. Typical representatives include GISCloud, CartoDB, ArcGIS Online, etc [1-2]. The current systems did a good job in mapping or data sharing, while rarely touched the area of geocomputation which sometimes is both compute- and data-intensive.

What happens when GIS meets HPC? Grid computing and cloud computing are two promising computational frameworks [3-5]. These researches have done explorative and solid work in this area, and tested on CyberGIS, XSEDE, Hadoop and Amazon EC2. The test results demonstrate that these platforms can provide high performance parallel computing capability. Parallel framework for processing massive spatial data is an active research topic recently. Researchers have proposed some parallel frameworks for point cloud algorithms [6] and remote sensing image processing web services [7]. However, these works pay little attention to combining geocompution with cartography. To promote the performance of computation in spatial analysis and visualization, we have been making great efforts in the past three years to build a GIS with full functions in terms of geospatial data management, visualization and especially, high-performance geospatial computing tool set. The name of the system is HiGIS which stands for high performance GIS. The backend of HiGIS is running on a HPC environment, while the frontend includes a desktop and a web client which is fairly thin and cross-platform. There are hundreds of parallel geospatial computing tools provided in HiGIS. Geospatial data, tools and geocomputation models are treated as the basic elements in the ecosystem around HiGIS.

In this paper, we depict a framework based on communicating sequential process (CSP) to support high performance computing for geographic information

This work was supported in part by the National Natural Science Foundation of China under Grants 41271403 and Grant 41471321, and the Natural Science Foundation of Hunan Province, China under Grant. 12jj4033.

Digital Object Identifier 10.4316/AECE.2015.03018

applications on the web. Parallel I/O strategy using file view was proposed to improve the performance of geospatial raster data service. In order to support web-based online mapping, an interactive cartographic script was designed to represent a map. This script acts as a map style specification sent to render service of HiGIS. All of these works have already been implemented as core services in this HPC-based GIS.

II. HiGIS: A High Performance GIS

The main goal of HiGIS is to improve the performance of time-consuming GIS operations by utilizing parallel computing in an HPC environment, as well as to provide such enhanced GIS ability to as many users as possible via personalized, light-weighted and cross-platform client programs [8].

A. Architecture

The server-side software stack of HiGIS includes system services supporting the three GIS core functions - spatial analysis, visualization and data management. Basic components of HiGIS contain a heavy server and a light client. Geospatial data storage and geocomputation toolbox are implemented on high-performance computing cluster. Fig. 1 shows three core GIS services, which are computing service (***higine***), cartography service (***hiart***) and repository service (***hipo***) forms full range of GIS services. The request of these services can be invoked via HTTP protocol by either desktop client or web client.

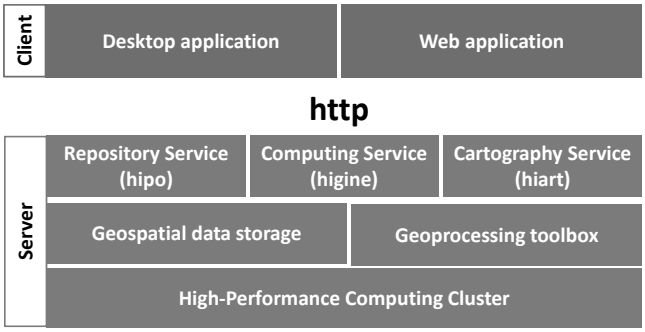


Figure 1. Core services of HiGIS

Higine is responsible for processing geospatial computing requests sent from client programs. In HiGIS, such kind of computing requests are normalized as workflows even the flow contains only one step of some fundamental analysis. ***higine*** can interpret and execute the submitted scripts. To ease the submission of geospatial computing requests from clients, ***higine*** defines a group of public data structures and service interfaces including submit, cancel, get_status, etc. Rather than directly forward the computing requests to the operating system, ***higine*** has to take care of the executing sequence of the submitted jobs, as well as how many computing resources should be assigned to each job because the underlying HPC environment is so different from a commodity personal computer running local operating system. Consequently, ***higine*** communicates with a batch job scheduler commonly used in HPC environments to schedule the computing resources and keep the load balanced.

Hiart provides interfaces for registering and styling geospatial data for cartographic visualization. Visualized geospatial data in HiGIS is served by Tile Map Service (TMS) standard. When a piece of geospatial data is visualized, its spatial reference system is normalized to Web Mercator system. The reason for this normalization is to avoid on-the-fly map re-projection.

Hipo encapsulates the access to the geospatial data repository and provides a unified interface to both the other server-side components and client programs. In the server-side, parallel I/O for raw raster data is used to improve I/O efficiency. When users import their own data to HiGIS, or export a piece of geospatial data to their local device, the data access interface will be invoked. The public service interfaces defined in ***hipo*** are mainly creation, retrieval, update and deletion (CRUD) operations of the data stored in the repository.

Fig. 2 describes how these core services serve together for applications. When a user submits a geocomputation job to ***higine***, ***higine*** decides how to dispatch the computing resources for the job according to the load of compute nodes. Once the geoprocessing tool is started to do the job, it sends a request to ***hipo*** to get geospatial data. If the job is finished, the result can be either saved by ***hipo*** or visualized by ***hiart***.

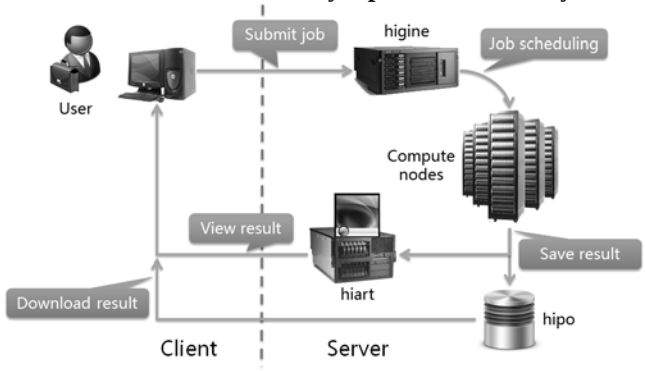


Figure 2. How to serve for applications with core services of HiGIS.

So far, the 1.0 version of HiGIS has already been released but is deployed in an internal HPC cluster with 32 compute nodes in National University of Defense Technology. The running system provides strong support to solve some large-scale geocomputation problems. For example, the job of watershed modeling for an area in the south of Fujian Province in China (about 60000 km^2) is finished within 5 minutes with 24 processes [9].

B. Object model

The concept of ***geoapp*** is proposed to serve as an abstract object model for the geospatial computing-related resources in HiGIS (Fig. 3). A ***geoapp*** object acts as a "producer", with some ***appoptions*** to determine what and how to produce. A ***map*** is a symbolic depiction highlighting relationships between elements of spatial entities, such as objects, regions, and themes. A ***geodata*** object represents a geospatial dataset with options like querying geographic extent to produce part of the dataset. A ***tool*** object represents an executable algorithm or a utility program, which produces the output or side effect of the execution. A ***model*** object is composed of simple ***geoapps***, e.g. ***geodata*** and tools. A ***model*** can produce composite results according to its definition and internal logics.

The usage of HiGIS is a little bit different from the traditional GIS. ***geoapp*** object model plays an important role

in the HiGIS user mode. First of all, an account for each user is necessary for the purpose of personalizing the ***geoapp*** resources owned by the user, as well as the working perspective of the system. HiGIS users may import their own ***geodata*** datasets to the system while taking full control of the access level of the data. Besides the user-imported data, HiGIS provides a large amount of free geospatial datasets acquired from other public data sources for high-performance geocomputation processing and research.

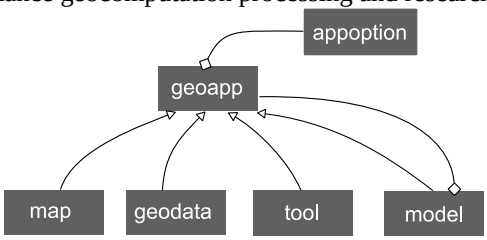


Figure 3. geoapp object model in HiGIS.

Then users may search and select the tools that fit for their processing requirements in the HiGIS tool repository. There is a group of research teams in China jointly doing research and development work for the HiGIS tool repository. The group has contributed to the initial high-performance spatial analysis tools taking great use of parallel computing powered by HPC. The toolbox contains from fundamental analysis tools like buffer and overlay analysis, digital terrain analysis, spatial data transformation, to advanced geocomputation tools like intelligent algorithms, geostatistics and even complicated watershed modelling, etc.

To launch a geospatial computing task, users can either directly submit their jobs by providing necessary parameters via the GUI of a tool, or construct a complicated HPC workflow with an embedded model designer. ***higine*** working at the server-side will take care of the processing of jobs and job flows. Users can monitor the detailed running information of their jobs in a dashboard. When the submitted geospatial computing task is completed, the result dataset can be visualized immediately if necessary.

If users want to browse the data stored in the system or the results generated by geocomputation tasks, they can add them to the map. HiGIS provides essential map operations such as pan, zoom and recenter. Moreover, users can adjust the order and visibility of map layers in HiGIS. These functions look as same as traditional GIS. The most attractive feature of HiGIS is that users can edit map styles on-the-fly based on the powerfully parallel rendering capability of ***hiart*** server. For example, to enhance the visual effect, users can finish the job of calculating relief amplitude of a raw SRTM dataset with cartographic styles.

C. Functional components

There are four basic functional components in HiGIS as shown in Fig. 4-6.

The first is Geospatial data manager, which implements: large-scale dataset storing, retrieving and previewing on-the-fly with the support of HPC, It provides feature identifying, selecting and measurement for vector data, and raster data attributes extracting, e.g. max/min/average value, histogram.

The second component is Cartographic editor, which make maps with comprehensive styles on the web, and support for massive layers corresponding to large-scale dataset.

The third component is Tool repository, which have parallel algorithms for registration, retrieval, execution, scheduling and monitoring.

Figure 4.Geospatial data manager

Figure 5. Cartographic editor

The fourth component is Model editor, which builds geocomputation workflow for high-flexibility. Even the map-making can be finished by building a workflow with a template cartography tool. All the tasks involved in the workflow will be scheduled properly in HPC environment.

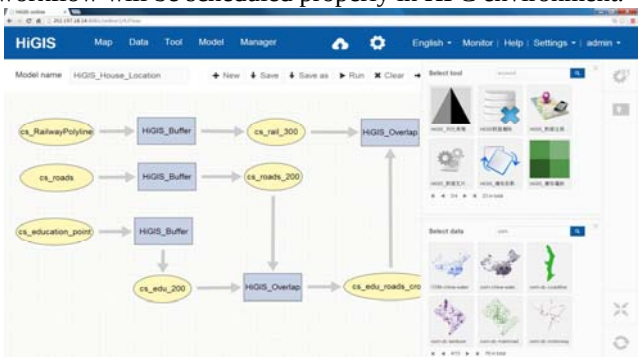


Figure 6. Model editor and Tool repository

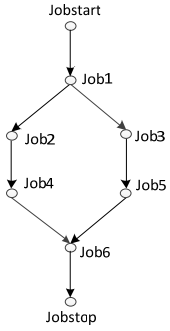
III. CSP Based Geocomputation Workflow

CSP is an algebraic theory proposed by Hoare [10] for solving complex problems caused by concurrency. There are several challenges when building a geocomputation workflow system over HPC clusters. The first and foremost, it should be easy to combine simple spatial analysis algorithms into complex solutions. Furthermore, parallel efficiency should be optimized at the workflow level, to fully exploit the potential of hardware and algorithms. The last but not the least, since HPC is costly, the system should offer some control abilities, like workflow cancelling and pausing/resuming, as well as a proper failure handling

mechanism. In this paper, a workflow system based on CSP is designed and developed targeted at these challenges.

A. Geocomputation workflow

There have been some researches explore the geocomputation framework based on cloud environment or MapReduce cluster. OpenRS-Cloud is a platform remote sensing image processing based on cloud computing technology [11]. MRGIS is a workflow system for GIS based on MapReduce clusters to ease geographic problems solving and achieved significant performance advantages compared to non HPC solution [12]. However, implementation details of MPI-based geocomputation workflow systems built on HPC clusters have not been investigated in depth [13]. As for fast visualizing the results of geocomputation, existing platforms and technology paid little attention.



The JSON structure of a workflow submitted to higine

```
{
    user: <string>,              //The user submitting workflow
    jobflow:{
        name: <string>,          //Name of workflow
        jobs_count: <int>,       //Number of jobs in the workflow
        jobs: [                  //List of the jobs
            {
                appid: <int>         //Id of job
                app_options: {       //Parameters of job
                    input: <data>,
                    output: <data>
                }
                runtime_context: { //Runtime environment
                    parallel: <int>,
                    mem: <int>
                }
                //Dependency betweent the jobs
                precursor_jobs_count: <int>,
                subsequent_jobs_count: <int>,
                precursor_jobs: <list>,
                subsequent_jobs: <list>
            }, ...
        ]
    }
}
```

Figure 7. DAG and its JSON specification of geocomputation workflow

As a complex processing procedure, a geocomputation workflow can be split into a bunch of steps. Each step in the workflow refers to a spatial analysis operation (action), mainly as a single algorithm with input and output geospatial dataset. The interfaces of operation are defined by a description specification along with the algorithm. The inputs of each step can be given either by users or former steps. States transition between actions represents the processing state of workflow. Directed acyclic graph (DAG) is used to model the workflow. DAG uses nodes and edges to illustrate the workflow corresponding to actions and data transfer between actions. Recursions or loops are not allowed in the graph to avoid validating complexity. A typical DAG of geocomputation workflow is shown in left part of Fig. 7. In this workflow, Jobstart and Jobstop are virtual nodes which denote the start and stop of workflow. Each directed edge denotes that the processed geospatial dataset transfer from one job to another. Precursor job and subsequent job represent the job executed before and after a job respectively. As an example, Job4 and Job5 are precursor jobs of Job6, and Job1 is subsequent job of Job2. A workflow can be created and submitted to higine using model editor of HiGIS. Model editor translates the DAG into JSON (JavaScript Object Notation) format, which is a commonly used data-interchange format in the web. The JSON specification designed in HiGIS is illustrated in the right part of Fig. 7.

To establish a flexible web geocomputation framework, some basic utility operations should be integrated. A bunch of workflow utility programs are developed in order to build reasonable geospatial applications, including metadata extracting, data publishing, web map service (WMS) publishing and other more. These tools can be easily and often used as actions in workflow.

Geospatial analysis and processing algorithms tend to be both computing intensive and communicating intensive, which restricts their scalability. Within the same workflow, computing resources could be balanced among steps considering the scalability of corresponding algorithms to raise both the resources utilization and the overall execution efficiency. Suppose there are two algorithms to run on a 256-cores cluster. Executing them one by one both with given 256 cores may cause low parallel efficiency. For most algorithms, the parallel efficiency of a 256-cores execution may be much lower than 128-cores execution in practice. Consequently, the workflow engine may optimize the resource assignment and execute the two algorithms simultaneously with both given 128 cores, which may bring better performance gains.

Stable workflow control facilities are rather difficult to implement since many real-world workflows are fairly complex and have many states, especially in a web environment. Concurrency must be carefully handled to avoid inconsistent states or deadlocks if synchronizations are heavily taken. Concurrent entities interact with each other all through explicit interfaces driven by an event-callback metaphor, in order to reduce uncertainty introduced by implicit interactions. When implementing web geocompution framework, some flexibility should be offered to handle failures in various ways. When a step fails, one may choose to either cancel all the unfinished steps in the workflow, or only cancel the steps dependent on the failed step. Both of the choices can avoid processing on the corrupted data.

B. CSP model for geocomputation workflow

In Fig. 8 there is a demonstration for two actions (rectangle) and six states (ellipse). The directed edges describe the state transitions of action changing between start, submit, suspended, awaked, canceled and finished. Even some states such as failed and revoked are omitted here. It can be easily observed that a composition explosion problem will be resulted in the case of multi-user concurrent environment. So DAG is suitable for describing workflow, but not proper for handling the concurrency problem in execution.

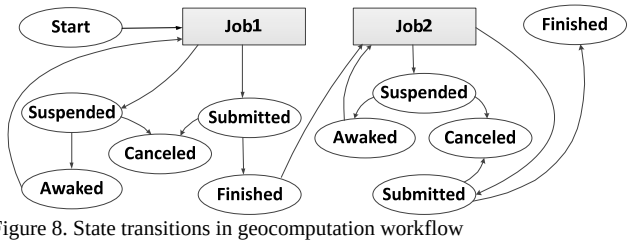


Figure 8. State transitions in geocomputation workflow

CSP uses message passing as the core concept, and defines a formal language to describe the interactions between different parts of a system. In CSP, a process represents the behavior pattern of an object we are interested in, associated with a finite set of possible events as its

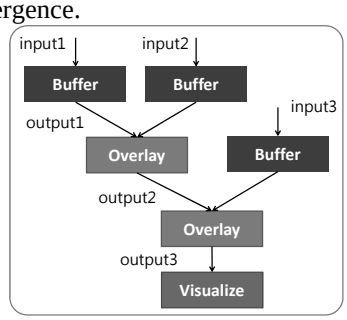
alphabet. Communication channels are defined upon these concepts using trace semantics. CSP uses uppercase letters P, Q, R to represent processes. Fundamental operators on processes are shown in Table I.

TABLE I. FUNDAMENTAL OPERATORS FOR GEOCOMPUTATION FLOW

Operator	Notation
Prefix	e -> P (e is an event)
Sequential composition	P ; Q
Parallel	P ‖ Q

In our application framework, each task, or equally, a running instance of an algorithm, is defined as a process. The workflow of an application which composed algorithms together will then be mapped to a CSP model, which could be reasoned to ensure correctness in certain tools. In the case of housing location calculation use as an example, the workflow can be described as: First, buffer areas are calculated on several kinds of interesting data. Second, these areas are overlaid using different operators. Finally the source data and result data will be visualized on a single map following a predefined style.

The sample geocomputation workflow is shown in left part of Fig. 9, and the CSP expression is listed in the right part. In HiGIS, this expression will be parsed by ***higine***, and submit to the HPC cluster. With CSP expression, some tools can be used to verify accuracy of workflow and guarantee robustness of ***higine*** [14]. By analyzing workflow models specified using CSP, these tools support automated model checking to keep executive away from deadlock or divergence.




```
Start->
(
  (
    (
      Buffer(input1)
      ||
      Buffer(input2)
    );
    Overlay(output1)
  )||
  Buffer(input3)
);
Overlay(output2);
Visualize(output3);
Skip
```

Figure 9. Example of geocomputation flow

When dealing with more complex real world problem, we can elaborate the model to fully describe the interactive pattern between processes. Based on the model, the parallel scheduling module will direct the data flow passing through every task, using multiple scheduling techniques to achieve high efficiency and robust concurrent control. Algorithms called by the jobs can be parallel ones using MPI, which can fully exploit the power of high performance clusters, and also can be sequential ones to be compatible with traditional spatial analysis libraries.

The core system of ***higine*** that parses and executes workflows is written in C++. The algorithms are mainly written in C++ based on MPI, while workflow utilities are written in various scripting languages like python, ruby, or bash. The system is open to non-MPI algorithms which, however, may not take advantages of special optimizations for MPI programs. The TORQUE resource manager and the MAUI scheduler are used for efficiently submit and execute MPI program [15-16]. OpenMPI is chosen as the MPI implementation while in most cases it is not a heavy work to switch to other implementations, e.g. MPICH2 or Intel MPI. Several geocomputation workflows are built for experiments. The experimental results showed the feasibility and efficiency of the workflow engine. It provides varies functions including executing algorithms in parallel, taking several previous outputs to produce new output, registration to metadata storage then published as Tile Map Service (TMS) automatically, and works well in HPC. Fig. 10 demonstrates the scheduling framework of ***higine***. The system consists of three components: ***higine***, TORQUE and MAUI. These components collaborate closely to serve for high performance execution and robust concurrent control of geocomputation.

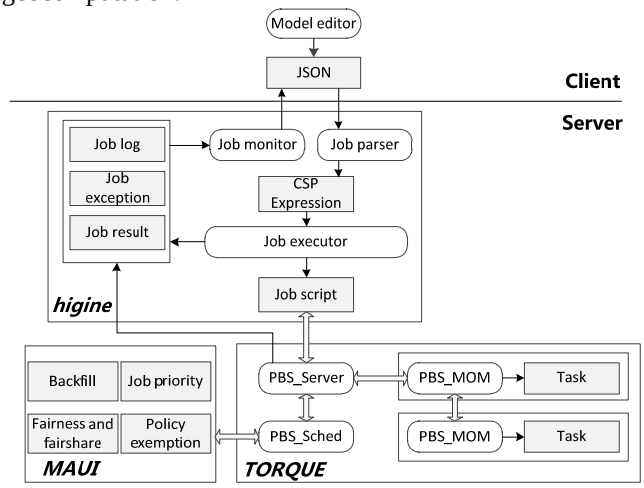


Figure 10. Scheduling framework of higine

For a specific workflow, the execution pattern is designed by model editor in client side. Then the workflow described using JSON specification send to server from the front end. In backend, job parser translated the JSON into CSP expressions. Job executor generates job scripts according to CSP expressions and starts multi threads to submit job scripts to TORQUE. The policy for choosing the best candidate execution nodes is provided by MAUI. MAUI is an extensible scheduler integrated some policies of "Backfill", "Job priority", "Fairness and fairshare" and "Policy exemption". The estimated completion time of each action in workflow is the most important factor for efficient scheduling. We have proposed a method for estimating completion time [17]. But so far, it can only apply to some specific algorithms. The running status and other information about the job such as log, exceptions and results, are recorded by job executor. Job monitor collects and reports these messages to users.

IV. PARALLEL DATA ACCESS IN HIGIS

High spatial resolution satellites collect petabytes of geospatial data from space every day, while citizen sensing activities are accumulating high temporal resolution data at a comparable or faster pace. These data are collected and archived every minute at various locations and record multiple phenomena of multiple regions at multiple scales. The geospatial data need to be calculated in spatial information applications is massively increasing. The complexity and the demand for accuracy in data processing are also growing. Geospatial data processing has presented more data-intensive and compute-intensive [6],[18]. Using multi-processor cluster and parallel computing technology

has become an inevitable trend. When the performance of processing is enhanced, the parallel computing processes would wait for data reading or writing if data access is serial. In this context, parallel data I/O is crucial for efficient geospatial data processing. Therefore, a parallel strategy for geospatial raster data is presented in this section.

A. Problem in Parallel Raster I/O Mode

There are usually two methods used in parallel processing geospatial raster data. The first is the Data Distribution and Collection (DDC) method as illustrated in Fig. 11. In this method, only the master process is responsible for all read and write operations of the geospatial raster data, while slave processes are responsible for data processing. The sending and receiving of data are implemented through an inter process message passing mechanism between the master and slave processes. The main drawback of the DDC method is that the master process easily becomes a bottleneck when the number of slave processes increases.

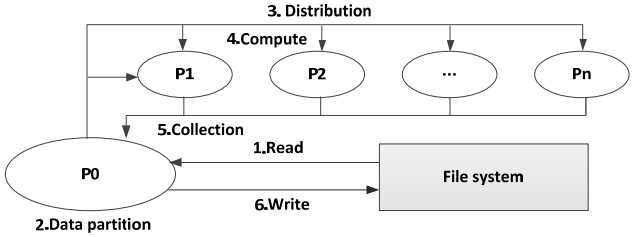


Figure 11. Parallel geospatial raster with distribution and collection

Another parallel I/O mode does not rely on the distribution and collection of the master process, as illustrated in Fig. 12. Each process can relatively independently access the data. Only Meta data is distributed by a master process. Each processes simultaneously access data, which can largely increase the overall I/O bandwidth. However, this approach requires an underlying parallel system support such as General Parallel File System (GPFS). Parallel file system provides high performance by allowing data to be accessed over multiple computers at once. Higher I/O performance can be gained by "striping" blocks of data from individual files over multiple disks, and reading and writing these blocks in parallel. Unfortunately, in a non-parallel system, e.g. Network File System (NFS), there are no strip blocks in the disks. If the read and write request very randomly, the file system must response to the requests randomly. Because of the mechanics of a disk transfer, in this situation, the I/O efficiency will be significantly reduced.

Some recent studies have tried to apply Geospatial Data Abstraction Library (GDAL) to parallel geospatial raster processing [19-20]. GDAL (http://www.gdal.org/) is a popular open source tool for geospatial raster I/O processing. These works explore the efficiency and flexibility of using GDAL in parallel raster I/O mode. Experimentations show that parallel raster I/O using GDAL cannot work well under column-wise or block-wise data partition. Two-phase I/O strategy is used to reduce the I/O requests through inter-process communication [20]. While small I/O requests combined into large contiguous I/O requests, the communication costs will likely be the new bottleneck. On the basis of a specific format (e.g. Hierarchical Data Format v5 [HDF5], Parallel Network Common Data Format [PnetCDF]), massive geospatial raster data set can be accessed by parallel I/O library [21]. But these specific formats will face the great challenge of diversity in many current geospatial applications. For the reason that we are living in the era of Big Data, large-scale of spatiotemporal data are generated per day. Consequently we cannot wait for transforming various formats of geospatial data to a specific format, and then processing these data.

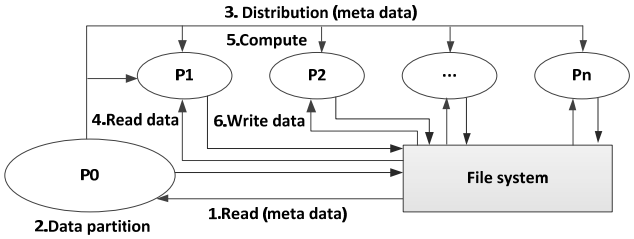


Figure 12. Parallel geospatial raster with distributing only metadata

B. Parallel Geospatial Raster I/O Using File View

MPI (Message Passing Interface) is a standardized message passing system designed for high performance computing. The standard defines a core of library routines use to develop portable and scalable large-scale parallel applications. This section introduces "file view" strategy which is suitable for both GPFS and NFS file system. File view is a new file concept introduced in MPI 2.0 standard. A process file view is created by calling up MPI-IO function. Each process can define its own file view made up of non-contiguous file segments. If the underlying MPI-IO implementation considers the access to these file segments as aggregated read/write call, the I/O performance should be greatly improved. This strategy was first implemented in MPI-IO for a much more efficient use of the I/O subsystem [22]. Based on file view mechanism, non-contiguous and piecemeal I/O requests are aggregated into a small amount of contiguous I/O requests. MPI-IO can be used to schedule the read/write sequences between multiple processes, and there is little data exchange in its implementation. File view is a critical optimization of parallel I/O, which allows communication of "big picture" to file system. Two-phase I/O strategy spends more communication costs on sending/receiving data between parallel processes. The basic idea of file view is building large blocks instead of using communication to precede I/O, so that I/O performance will be largely improved.

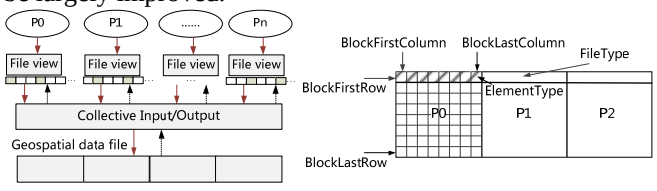


Figure 13. Parallel geospatial raster I/O using file view

Fig. 13 shows the basic process of parallel geospatial raster I/O using file view. Firstly, all processes read metadata of the geospatial raster data to be processed using GDAL. These information is stored in a memory data structure, where includes: MPI file handle, columns and rows of raster data cells, bands of raster data, data types of cell, absolute offset address of raster data in the file. After that each process calculating the data size and offsets needed to read and process. Data partitioning methods can be row-,

column-, or block-wise. Subsequently, one specific process is responsible for creating the output file with the metadata. Once created, the process will broadcast to other processes. Other processes start to read raster data to be processed. Finally, each process completes their computing tasks, and then opens the output file, sets the respective file view and writes the result data into the output file.

The example of geospatial data file view is illustrated as the right part of Fig. 13. Assuming data partitioning method is block-wise. File view includes three elements: absolute offset address (*Displacement*), the basic element type (*ElementType*) and file type (*FileType*). There are n parallel processes (P0, P1, … Pn). The rows and columns of geospatial raster cell is *RasterYSize* and *RasterXSize*, and cell data type is *ElementType*. For process P0 the file view parameters are: the *RowB1*, *RowBn*, *ColB1* and *ColBn* are used to compute the block sizes of the raster data for P0 to process. The row size of a block is

$$BlockYSize = RowBn - RowB1 \quad (1)$$

and column size of a block is

$$BlockXSize = ColBn - ColB1 \quad (2)$$

The *FileType* is made up of *ElementType* (*BlockXSize*), hole (*RasterXSize* - *BlockXSize*), and one *ElementType* in Y dimension. After setting the file type, each process can use its own file view to read and write data. We can use this file type in MPI_File_set_view and MPI_File_write_at_all functions provided by MPI-IO to implement the parallel I/O. It should be noted that file view type setting is depended on different MPI version. We can only support GeoTiff format as an output type so far. But it is easy to extend our function interfaces to support other geospatial raster formats.

TABLE II. FUNCTION INTERFACES FOR FILE VIEW

Name	Description
create_raster	Create a geospatial raster file to read or write
open_raster	Open a specified geospatial raster file to read/write
close_raster	Close a opened geospatial raster dataset
write_rows	Data partition is row-wise. Write data rows
write_cols	Data partition is column-wise. Write data columns
write_blocks	Data partition is block-wise. Write a data block

We defined a set of function interfaces to encapsulate the operations to parallel access raster data, including create_raster, open_raster, close_raster, write_rows, write_cols and write_blocks listed in Table II.

V. INTERACTIVE CARTOGRAPHIC SCRIPT

The OpenStreetMap project provides user-generated street maps [23], which make it easier for crowdsourcing cartography. While traditional mapping is often carried out by professional organizations, crowdsourcing cartography generates a map using informal social networks and web 2.0 technologies. Mapbox (http://www.mapbox.com) offers commercial mapping services using Python scripts for the data parsing, CartoCSS for the designing of the maps, JavaScript to make the maps interactive, and HTML to combine all the layers together a set of maps were created. By this means, anyone can design a map and choose what he wants on the map and exactly how it looks. Every detail is under control, e.g., adding data, picking fonts and colors.

A. Interactive cartography in HiGIS

Interactive cartographic script can be used to define the style of a map, which enables the interpretation of the map, the construction of page layouts. These scripts need to be parsed as actual cartographic instructions sent to a map renderer. According to Web Map Service standard proposed by Open Geospatial Consortium, Map can be defined as follows:

$$Map = \left\{ \left(Layer_i, Style_i \right) \middle| 0 < i < N \right\} \quad (3)$$

Where $Layer_i$ is a layer in a map, and $Style_i$ is a text defining the map elements of $Layer_i$, such as legends, titles, and related text or symbols. Cartographic script designed in HiGIS includes two parts, Selector and Declaration. Selector defines the subject for map manipulation, such as map, layer, and objects with filter conditions. The declaration defines the style of the subject. The syntax can be described as follows.

```
Selector {
  Declaration;
  Selector {
    Declaration;
  }
}
```

As an example of a map, if there is a highway layer graded to different classes, and we want to render the class-1 highway into the red color. Therefore the Selectors are highway layer and class-1 highway, Declaration is "line-color=#ff0000". This map can be defined as follows.

```
# highway {
  [ class=1] {
    line-color = #ff0000;
  }
}
```

Interactive cartography in HiGIS includes three steps as shown in Fig. 14. The user sends a request to the Register to create a map. Then CartoCSS scripts specifying map layers and styles are sent to the Parser. Parser complies CartoCSS scripts to scripts which can be processed by Renderer. Borrowing HPC power, Renderer generates the map tiles in parallel. User can view a map by sending the request to the Tiles server. If the user is not satisfied with the results, he can re-execute these steps to adjust.

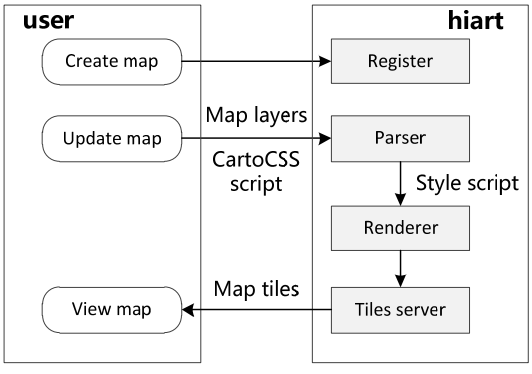


Figure 14. Interactive cartography in HiGIS

B. Mapping with template

The cartographic script gave plenty of control to the person writing the script, but the user experience was not very friendly. The scripts had to be written as text files. Even drawing a simple map required writing a somewhat lengthy script. Many users were desired to simplify these

works. HiGIS provided automation capability for mapping from template.

A sample use case of HiGIS contains data selection, spatial analysis and cartography. Assuming we want to buy a house in a city. An eligible housing area should have a good surrounding environment and provide with convenient living conditions. Therefore, we need to take into account the city's education, transportation and other important factors: such as, schools, shopping malls and hospitals are in the housing area within 100 meters, railways, main roads must be 200 meters away from housing area, and housing area is less than 100 meters to district roads, and so on. It is a typical application of workflow shown in Fig. 9.

According to the workflow described above, typical use mode in HiGIS is as below:

1. Construction of the geocomputation flow model for housing locating area.

2. Select input geospatial data set such as points of interest, railways and roads et al shown in Fig. 15. These data are rendered in default map styles.

3. Run the model, and each step in the execution progress can be observed dynamically.

4. Monitor the status of the task (I/O time and computing time), run time and results in the dashboard.

5. If the task is finished, open results into the map, the housing areas can be visualized in a comprehensive style as shown in Fig. 15.

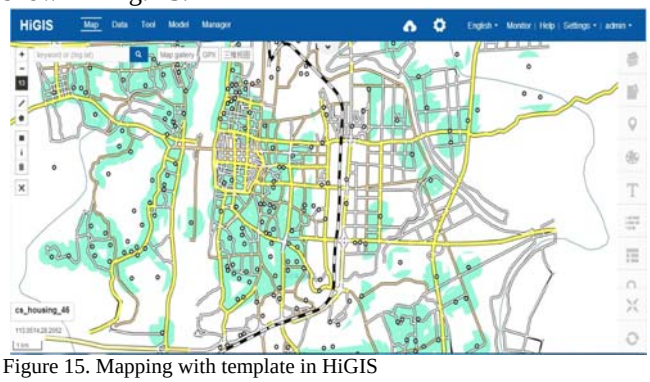


Figure 15. Mapping with template in HiGIS

The resulted map is made from a template. A map template is described by cartographic script. When mapping with template, a new map is forked from an existing template. Then the layers of the new map are replaced according to result data. In order to generate a map as the same style as a template, the properties of layers in the new map must be consistent with those in template, which means the number of layers, order of layers, and attribute names of each layer must be the same.

VI. Experiment and Results

A. Experimental environment and datasets

The experiment environment was an IBM SMP cluster with 32 server nodes running RedHat Enterprise Linux 6.4. Each server node consists of two Intel Xeon (X5540 2.4GHz) six-core CPUs and 24 GB DDRIII memory. The role assigned to server nodes is: four nodes for data service (d01-d04), four nodes for visualization service (v01-v04), and 28 nodes for computing service (c01-c28). Test datasets are listed in Table III. Geometry features are stored in PostGIS. Raster data are stored in GeoTiff files on 24 shared disks (3TB each disk) which is used as storage array. General Parallel File System (GPFS) is used for sharing test data between compute nodes, and configured with default parameters. The network configuration is 10GigE.

Table III. Test datasets

Notation	Type	Description
Data1	Raster	Dimension: 31,250*34,472, cell depth: 1 Byte
Data2	Raster	Dimension: 16,160*17,128, cell depth: 2 Byte
Data3	Raster	Dimension: 8,290*9,598, cell depth: 2 Byte
Data4	Feature	Road network of Beijing, feature count: 71,862
Data5	Feature	Road network of Beijing, feature count: 72,308
Data6	Feature	Point data of Beijing, feature count: 251,282
Data7	Raster	DEM data, Dimension: 30,000*30,000
Data8	Raster	DEM data, Dimension: 6,000*6,000

For the purpose of parallel performance comparison to ArcGIS, we deploy HiGIS and ArcGIS 10.2 Desktop in the same hardware environment. The server is a SuperMicro server with four Intel Xeon E5-4620 processors, clocked at 2.2 GHz, and 512 GB memory of DDR3-1333. ArcGIS is installed on Windows 7.

B. Parallel performance

In order to measure the parallel performance of workflow, the jobs are submitted in sequential and parallel mode in HiGIS. The case study of locating the house is used to examine the parallel performance. Test datasets are Data4 to Data6. The results are listed in Table 4. It is clear to see that Buffer1, Buffer2 and Buffer3 are simultaneously started. When Buffer1 is ended, computing nodes c01, c02 and c03 are released for Overlay1. The same situation can be seen in Overlay2 and Visualization. In parallel mode, the runtime of all jobs is 1 minutes 18 seconds. While in sequential mode, the runtime of all jobs is 2 minutes 22 seconds. The experimental results are similar in IBM SMP cluster and SuperMicro server. When using ArcGIS ModelBuilder to submit the same workflow with the same datasets, the results are gained after 2 hours running. Because spatial analysis operations are parallelized and the workflow is also scheduled by ***higine*** in parallel, there is two-level parallel processing in HiGIS. Thus, it shows the parallelization performance of HiGIS through the case study.

Table IV. Parallel performance of workflow

Job	Node	Start	End	Runtime(s)
Buffer1	c01-c03	00:00:00	00:00:21	21
Buffer2	c04-c07	00:00:00	00:00:25	25
Overlay1	c01-c03	00:00:25	00:00:47	22
Buffer3	c08-c10	00:00:00	00:00:23	23
Overlay2	c01-c03	00:00:47	00:01:15	28
Visualization	v01-v03	00:01:15	00:01:18	3

Parallel I/O performance is evaluated by measuring the MB/s to read and write data from/to a raster file with/without file view. The data is partitioned by block-wise decomposition. The block size is depended on the number of processes. In the experiment, we compared the performance to the method without file view mechanism, and also the GDAL-2P strategy [20]. As Fig. 14 shown, if parallel processes access the data file (Data1) without file view, I/O performance will be significantly lower than the aggregate request method using file view. Two-phase I/O strategy GDAL-2P performs better but still lower than file view method thanks to communication costs. When increasing the number of processes, without file view, parallel I/O

performance will decrease with the increase of the number of processes. When the number is less than 32, I/O performance remained stable using file view. It can be observed that the increase in performance up to 6 processes followed by a decline with file view. This is because when the number of processes is increasing, the costs they take to schedule, communication and other system overhead also increased. The more number of processes is used, said the size of the blocks for decomposition is smaller. Therefore, the experimental results also reflect the influence of block size on the overall performance.

The parallel I/O performance shows some correlation to the size of data as Fig. 16-17. Generally speaking, the larger the data size, I/O performance is better. Meanwhile, when increasing the number of processes, dip in performance is later seen in larger data.

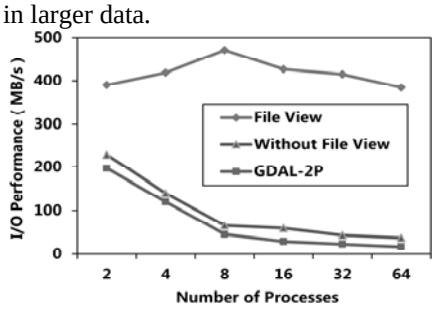


Figure 16. Performance comparison of different I/O methods

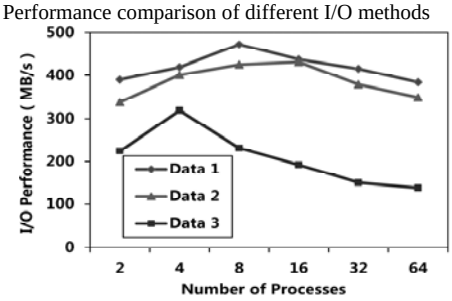


Figure 17. Performance comparison between different datasets

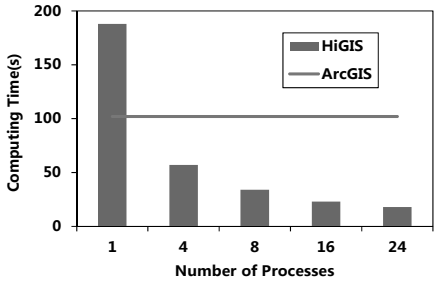


Figure 18. Performance of parallel slope analysis on Data7

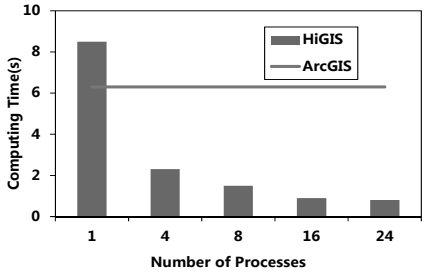


Figure 19. Performance of parallel slope analysis on Data8

In Fig. 18-19, there are the parallel performance comparisons of slope analysis between HiGIS and ArcGIS. The test datasets are Data7 (Fig. 18) and Data8 (Fig. 19). It can be observed that despite the size of test data, HiGIS outperforms ArcGIS when the number of processes is more than 4. ArcGIS Desktop supports multiple processors/cores with parallel processing, and performs extremely better than HiGIS when using 1 CPU and 4 processes especially for large size data (Data7). It suggests that ArcGIS Desktop can get the exclusive use of one CPU. However, when increasing the number of processes, ArcGIS performs comparably. HiGIS allows users to process geocomputation tasks cross multiple processors. By doing so, we can achieve more than 40% parallel efficiency for slope analysis.

C. Concurrency performance

Tsung (http://tsung.erlang-projects.org/) is a distributed load testing tool. Tsung's main strength is its ability to simulate a huge number of simultaneous users from a single machine. We can easily set-up and maintain an impressive load on a server. Tsung 1.4.2 is used to test the scalability and performance of core services of HiGIS. We tested the load and stress of ***hiart***, ***hipo***, and ***higine***. The HTTP response results of HiGIS are shown in Fig. 20-23.

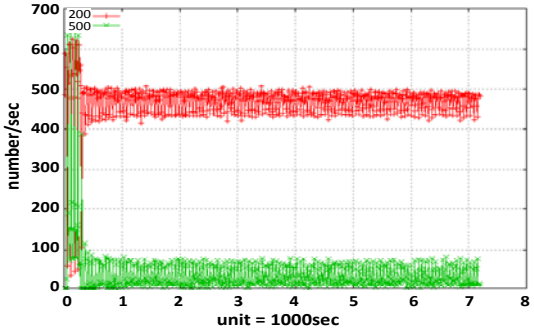


Figure 20. Concurrency test for meta data service of hipo

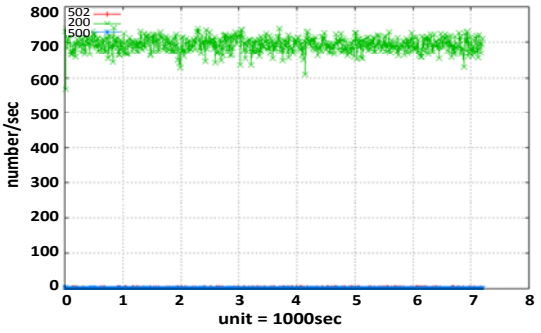


Figure 21. Concurrency test for spatial data service of hipo

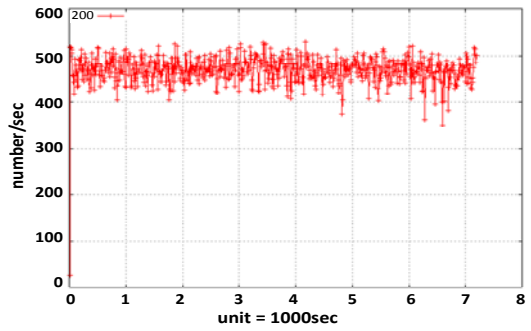


Figure 22. Concurrency test for hiart

The successful metadata query requests are more than 400 per second, and successful spatial query requests are more than 700 per second. Successful cartography requests are more than 400 per second, Average response time for 20

users to submit 2000 tasks is less than 0.1 second. These results prove that HiGIS is efficient and scalable.

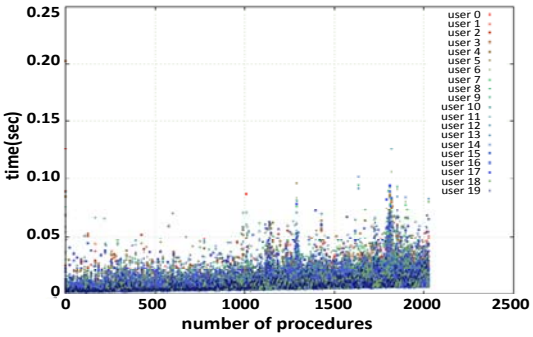


Figure 23. Concurrency test for higine

VII. Conclusion

In this paper, we propose a framework based on HPC and CSP to enable quick building of decent geographical information applications in the web. The prototype system HiGIS attempts to reduce the complexity when building efficient solutions to complex geographical problems. Taking house location calculation as a use case, we demonstrate that HiGIS can easily build a spatial analysis application integrated with a full stack of GIS tools like spatial analysis algorithms, geographical data management, spatial query and cartography, which take full advantages of high performance computing. The experimental results show both stability due to CSP and efficiency due to HPC, which proves the feasibility of the system.

In the near future, the visualization and processing component of HiGIS will be more compatible with various types of data, e.g. spatio-temporal data, 3D data and user-defined data. Spatial database in main memory is considered to improve parallel I/O for geometry features data. In addition, an open development framework including mapping, processing and data access API will be provided to enable ***geoapp*** development based on HiGIS, which will help form an ecosystem around HiGIS.

Acknowledgment

We deeply appreciate the supports from other research groups in China including Institute of Geographic Sciences and Natural Resources Research in C.A.S., Nanjing Normal University, Wuhan University, Nanjing University and North East University, for their contribution to the HiGIS tool repository.

References

[1] A. G. Aly and N. M. Labib, "Proposed Model of GIS-based Cloud Computing Architecture for Emergency System," Int. J. Comput. Sci., vol. 1, no. 4, pp. 17-28, 2013.

[2] J. de la Torre, "Organising geo-temporal data with CartoDB. an open source database on the cloud," In Proc. Biodiversity Informatics Horizons, Rome, Italy, Sept. 2013

[3] S. Wang, "CyberGIS: blueprint for integrated and scalable geospatial software ecosystems," Int. J. Geogr. Inf. Sci., vol. 27, no. 11, pp. 2119-2121, 2013. doi: 10.1080/13658816.2013.841318

[4] I.H. Kim and M.H. Tsou, "Enabling Digital Earth simulation models using cloud computing or grid computing-two approaches supporting high-performance GIS simulation frameworks," Int. J. Digit. Earth, vol. 6, no. 4, pp. 383-403, 2013. doi: 10.1080/17538947.2013.783125

[5] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang, and J. Saltz, "Hadoop gis: a high performance spatial data warehousing system over mapreduce," Proc. VLDB Endow., vol. 6, no. 11, pp. 1009-1020, 2013. doi: 10.14778/2536222.2536227

[6] X. Guan, H. Wu, and L. Li, "A Parallel Framework for Processing Massive Spatial Data with a Split-and-Merge Paradigm," Trans. GIS, vol. 16, no. 6, pp. 829-843, 2012. doi: 10.1111/j.1467-9671.2012.01347.x

[7] W. Guo, X. Zhu, T. Hu, and L. Fan, "A Multi-granularity Parallel Model for Unified Remote Sensing Image Processing WebServices," Trans. GIS, vol. 16, no. 6, pp. 845-866, 2012. doi: 10.1111/j.1467-9671.2012.01367.x

[8] L. Liu, A. Yang, L. Chen, W. Xiong, Q. Wu, and N. Jing, "HiGIS - When GIS Meets HPC," In Proc. 12th Int. Conf. on GeoComputation, WuHan, 2013. [Online]. Available: http://www.geocomputation.org/2013/papers/26.pdf

[9] J. Liu, A.X. Zhu, Y. Liu, T. Zhu, and C.Z. Qin, "A layered approach to parallel computing for spatially distributed hydrological modeling," Environ. Model. Softw., vol. 51, no. 0, pp. 221 - 227, 2014. doi: 10.1016/j.envsoft.2013.10.005

[10] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe, "A Theory of Communicating Sequential Processes," J ACM, vol. 31, no. 3, pp. 560-599, Jun. 1984. doi: 10.1145/828.833.

[11] W. Guo, J.Y. Gong, W.S. Jiang, Y. Liu and G. She, "OpenRS-Cloud: A remote sensing image processing platform based on cloud computing environment," Sci. CHINA Technol. Sci., vol. 53, no. 1, pp. 221-230, 2010. doi: 10.1007/s11431-010-3234-y

[12] Q. Chen, L. Wang, and Z. Shang, "MRGIS: A MapReduce-Enabled High Performance Workflow System for GIS," in Proc. of the 2008 Fourth IEEE Int. Conf. on eScience, Washington, DC, USA, 2008, pp. 646-651. doi: 10.1109/eScience.2008.169

[13] Y. Ma, D. Liu and J. Li, "A new framework of cluster-based parallel processing system for high-performance geo-computing," In Geoscience and Remote Sensing Symposium, Cape Town, 2009, vol. 4, pp. IV49-IV52. doi: 10.1109/IGARSS.2009.5417598

[14] T. Yuan, Y. Tang, X. Wu, Y. Zhang, H. Zhu, J. Guo, and W. Qin, "Formalization and Verification of REST on HTTP Using CSP," Electron. Notes Theor. Comput. Sci., vol. 309, no. 0, pp. 75-93, 2014. doi: 10.1016/j.entcs.2014.12.007

[15] G. Staples, "TORQUE Resource Manager," in Proc. of the 2006 ACM/IEEE Conf. on Supercomputing, New York, NY, USA, 2006. doi: 10.1145/1188455.1188464

[16] D. Jackson, Q. Snell, and M. Clement, "Core Algorithms of the Maui Scheduler," in Job Scheduling Strategies for Parallel Processing, vol. 2221, D. Feitelson and L. Rudolph, Eds. Springer Berlin Heidelberg, 2001, pp. 87-102. doi: 10.1007/3-540-45540-X_6

[17] S. Zhang, L. Chen, W. Xiong, "Research on performances of parallel programming models based on chip multi-processor," in Proc. 2011 Int. Conf. Computer Application and System Modeling, XiaMen, 2011, pp. 2688-2691.

[18] C. Yang, M. Goodchild, Q. Huang, D. Nebert, R. Raskin, Y. Xu, M. Bambacus, and D. Fay, "Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing?," Int. J. Digit. Earth, vol. 4, no. 4, pp. 305-329, 2011. doi:10.1016/j.cageo.2012.04.021

[19] L. Ouyang, J. Huang, X. Wu, and B. Yu, "Parallel Access Optimization Technique for Geographic Raster Data," in Geo-Informatics in Resource Management and Sustainable Ecosystem, vol. 398, F. Bian, Y. Xie, X. Cui, and Y. Zeng, Eds. Springer Berlin Heidelberg, 2013, pp. 533-542. doi: 10.1007/978-3-642-45025-9_52

[20] C.Z. Qin, L.J. Zhan, and A.X. Zhu, "How to Apply the Geospatial Data Abstraction Library (GDAL) Properly to Parallel Geospatial Raster I/O?," Trans. GIS, vol. 18, no. 6, pp. 950-957, 2014. doi: 10.1111/tgis.12068.

[21] Y. Zou, W. Xue, and S. Liu, "A case study of large-scale parallel I/O analysis and optimization for numerical weather prediction system," Future Gener. Comput. Syst., vol. 37, no. 0, pp. 378-389, 2014. doi: 10.1016/j.future.2013.12.039

[22] R. Thakur, W. Gropp, and E. Lusk, "Optimizing noncontiguous accesses in MPI-IO," Parallel Comput., vol. 28, no. 1, pp. 83 - 105, 2002. doi: 10.1016/S0167-8191(01)00129-6

[23] C. Heipke, "Crowdsourcing geospatial data," ISPRS J. Photogramm. Remote Sens., vol. 65, no. 6, pp. 550-557, 2010. doi: 10.1016/j.isprsjprs.2010.06.005