Environmental Modelling & Software 96 (2017) 347–360

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

pRPL + pGTIOL: The marriage of a parallel processing library and a parallel I/O library for big raster data

Jinli Miao [a, b], Qingfeng Guan [a, c, *], Shujian Hu [a, c]

[a] School of Information Engineering, China University of Geosciences, Wuhan, Hubei 430074, China
[b] Development and Research Center, China Geological Survey, Beijing 100037, China
[c] National Engineering Research Center of GIS, China University of Geosciences, Wuhan, Hubei 430074, China

ARTICLE INFO

Article history:
Received 13 August 2016
Received in revised form
2 April 2017
Accepted 19 June 2017

Keywords:
Raster
Parallel computing
Parallel I/O
Programming library

ABSTRACT

Data I/O has become a major bottleneck of computational performance of geospatial analysis and modeling. In this study, a parallel GeoTIFF I/O library (pGTIOL) was developed. Through the storage mapping and data arrangement techniques, pGTIOL can operate on files in either strip or tile storage mode, read/write any sub-domain of data within the raster dataset. pGTIOL enables asynchronized I/O, which means a process can read/write its own sub-domains of data when necessary without synchronizing with other processes. pGTIOL was integrated into the parallel raster processing library (pRPL). Several pGTIOL-based data I/O functions and options were added to pRPL, while the existing functions of pRPL stay intact. Experiments showed that the integration of pRPL and pGTIOL achieved higher performance than the original pRPL that uses GDAL as the I/O interface. Therefore, pRPL + pGTIOL enables transparent parallelism for high-performance raster processing with the capability of true parallel I/O of massive raster datasets.

Software availability

The software is still under test, but available for download at https://github.com/HPSCIL/pRPL. The demonstration program (i.e., pAspect) can be compiled by following the instructions in README.

1. Introduction

In the Big Data era, massive volume of geospatial data is being collected, generated, and accumulated at an unprecedentedly rapid pace, partially attributed to the fast and wide adoption of location-aware sensing technologies, such as satellite/aerial imaging systems, web cam, mobile phones, and various sensors. As results, a number of large geospatial datasets with regional/global coverages and high resolutions have been created, such as the 30-m ASTER Global Digital Elevation Model (ASTER GDEM, Tachikawa et al., 2011) and the 30-m Global Land Cover Dataset (GlobeLand30, Chen et al., 2015). Meanwhile, many complicated spatial algorithms and models have been developed in the last few years to analyze and simulate complex geospatial relationships, interactions, and dynamics. The increasing consciousness and interests of socio-economic globalization and global climate change have stimulated the use of regional/global geospatial datasets and complicated spatial analytical and modeling techniques to deal with large-scale problems and provides decision-making supports. However, the extremely high data intensity and computational intensity often demand for vast memory space and extensive computing time, making such geospatial applications inefficient and unscalable, sometimes even infeasible.

The last decade has seen a booming rise of studies and applications of high-performance computing (HPC) in geospatial studies, such as land-use modeling (Li et al., 2010; Pijanowski et al., 2014), terrain analysis (Qin and Zhan, 2012; Zhao et al., 2013), and geostatistics (Guan et al., 2011; Shi and Ye, 2013). The recent developments of CyberGIS (Wang and Armstrong, 2009; Wang, 2010), spatial cloud computing (Yang et al., 2011, 2013), and graphics processing units (GPUs) also stimulate the deployment of parallel computing in geospatial studies (e.g., Tang et al., 2011; Shook et al., 2013; Huang et al., 2013; Liu et al., 2013; Zhang and You, 2013; Shi and Ye, 2013), as they provide easy-to-access HPC facilities and platforms.

* Corresponding author. School of Information Engineering, China University of Geosciences, Wuhan, Hubei 430074, China.
E-mail address: guanqf@cug.edu.cn (Q. Guan).

http://dx.doi.org/10.1016/j.envsoft.2017.06.031
1364-8152/

While parallel computing has been proved to be a promising solution to the computational barrier induced by massive datasets and complex algorithms, the complexity of developing parallel applications remains high and has become one of the major obstacles to the wide adoption of parallel computing in geospatial studies. A few efforts have been made to reduce the development complexity of parallel geospatial applications. Cheng et al. (2012) developed a general-purpose optimization methods for the parallelization of digital terrain analysis based on cellular automata, which can also be used to parallelize other types of spatial analysis. Qin et al. (2014) developed a set of parallel raster-based GeoComputation operators (PaRGO) for users to implement parallel geospatial applications on three types of parallel computing platforms: GPU supported by compute unified device architecture (CUDA), Beowulf cluster supported by message passing interface (MPI), and symmetrical multiprocessing cluster supported by MPI and open multiprocessing. Shook et al. (2016) developed a parallel cartographic modeling language (PCML), a domain-specific language implemented in Python that is capable of automatically executing an user-written script in parallel.

The parallel Raster Processing Library (pRPL) is an open-source, general-purpose programming library designed to facilitate GIS scientists and professionals to develop parallel geospatial applications with minimal knowledge and skills of parallel computing (Guan, 2009; Guan and Clarke, 2010; Guan et al., 2014). By encapsulating the complex parallel computing details and providing easy-to-use interfaces, pRPL enables transparent parallelism such that users only need to focus on their own raster processing algorithms, and are able to implement parallel applications just like writing sequential programs. However, since pRPL was originally designed to facilitate the development of parallel raster processing procedures, data I/O was not one of the main focuses. pRPL 1.0 uses a centralized I/O mechanism and only provides a primitive method for reading and writing raster data in a specific ASCII format. pRPL 2.0 improves the I/O capability by incorporating the Geospatial Data Abstraction Library (GDAL, http://www.gdal.org) as the I/O interface to support a large variety of raster formats. Besides the centralized I/O mode, pRPL 2.0 also provides a pseudo parallel I/O mode. Nevertheless, the pseudo parallel writing mechanism in pRPL 2.0 generates multiple temporary files before combining them into the final output dataset. Therefore it is not true parallel I/O and the performance is sometimes even poorer than the centralized I/O when writing a large quantity of data (Guan et al., 2014). Thus, a true parallel I/O mechanism is needed to allow multiple subsets of data to be read or written concurrently without any intermediate files.

In this study, we developed a parallel raster data I/O library (pGTIOL) that implements a true parallel I/O mechanism for the GeoTIFF format, one of the most commonly used geospatial raster formats. We also integrated pGTIOL with pRPL to improve the I/O performance of pRPL. Several pGTIOL-based data I/O functions and options were added to pRPL, while the existing functions of pRPL stay intact. To examine the performance of the pRPL + pGTIOL integration, a parallel program for calculating slope and aspect was developed and a series of experiments were conducted. The results showed that pGTIOL greatly helped improve the I/O performance, and yielded better performance than the GDAL-based I/O methods of the original pRPL.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to pRPL 2.0. Section 3 elaborates on pGTIOL, including its design and implementation. Section 4 describes the integration of pRPL and pGTIOL. Section 5 presents the demonstration application and experiments, as well as the performance assessments. Conclusion is given in section 6.

2. pRPL 2.0

pRPL is a general-purpose programming library, specifically designed for GIS scientists and professionals who lack of knowledge and skills of parallel computing, to easily parallelize their own application-specific raster-processing algorithms and models. pRPL is based on the Message Passing Interface (MPI, Gropp et al., 1998) and written in C++, both of which are supported by a wide range of HPC architectures (e.g., computer clusters, massive parallel computers, and desktop computers with multi-core CPUs). Therefore it guarantees the portability of parallel applications across various HPC systems. pRPL-based applications can also be deployed in cloud computing environments, as long as a parallel computing platform with multiple CPU cores can be virtualized.

The rest of this section gives a brief introduction of pRPL 2.0, including its key components, features, and programming model.

2.1. Key components of pRPL

The most commonly used strategy for parallelizing raster-processing algorithms is data parallelism, which divides a grid of cells (i.e., domain) into multiple sub-grids (i.e., sub-domains) and utilizes multiple computing units (e.g., CPUs or CPU cores) to apply the same computational procedure on these sub-domains simultaneously (Fig. 1). Task parallelism, on the other hand, divides a task into a set of sub-tasks and applies them on data concurrently. Compared with task parallelism, data parallelism is easier to implement and suited for a wide variety of raster operations, including local, focal, zonal, and global ones, as long as the computation on a sub-domain is independent from the computation on others. Note that computational independence does not mean data independence. Some raster operations, while being parallelizable, may need data from other sub-domains when applied on a certain sub-domain. Typical examples are focal operations, also termed neighborhood-scope or moving-window operations. In order to provide general support for various user-defined algorithms, pRPL was designed primarily based on the principles of data parallelism. In other words, an application-specific algorithm (termed *Transition*), as long as its computational procedure is parallelizable in a data parallelism manner, can be easily parallelized using pRPL.

To construct a data parallelism framework for raster processing, pRPL includes a hierarchy of data containers, i.e., *Cellspace, SubCellspace,* and *Layer* (Fig. 2). A data container includes a *CellspaceInfo* attribute component indicating its dimensions (numbers of rows and columns), data type, data size, and NoData value. Also, a *CellspaceGeoinfo* attribute component is used to indicate the spatial reference information, including the projection, cell size (in a geospatial measurement unit), and geospatial coordinates of the northwest corner.

The main purpose of data containers is to hold the data to be processed. A *Cellspace* object, representing a domain, contains a matrix of cell values. The *SubCellspace* class is a child class derived from *Cellspace*, with an additional *SubCellspaceInfo* attribute component indicating the *SubCellspace*'s ID, dimensions, and minimal bounding rectangle (MBR) within a *Cellspace*. Thus, a *SubCellspace* object, representing a sub-domain, contains a subset of cell values in a whole *Cellspace*. A *Layer* object may include a whole *Cellspace* and/or multiple *SubCellspaces* (Fig. 1).

To support focal operations, the *Neighborhood* class can represent various configurations (shapes and weight distributions) of a moving window, including the regular Von Neumann and Moore neighborhoods, and user-defined irregular, asymmetric and discontinuous ones.

The *DataManager* class provides integrated data management