pRPL 2.0: improving the parallel raster processing library

Qingfeng Guan, Wen Zeng, Shishi Liu

Faculty of Information Engineering
China University of Geosciences (Wuhan)
Wuhan, Hubei, China
guanqf@gmail.com

The parallel Raster Processing Library (pRPL, http://sourceforge.net/projects/prpl/) is an open-source programming library to enable GIS scientists and professionals with basic programming skills but who lack knowledge and experience of parallel computing and programming, to easily develop and implement application-specific raster-based algorithms, analytical procedures, and models (Guan and Clarke 2010). As a generic parallel raster-processing library, pRPL supports both centralized (i.e., only the central cell of a neighbourhood can be updated during the process) and non-centralized (i.e., any cell can be updated) algorithms, therefore it supports not only local-scope and neighborhood-scope processes, but also some regional-scope and global-scope ones. For neighborhood-scope processes, pRPL supports any configuration of neighbourhood, including regular von Neumann, Moore, and extended Moore neighborhoods, as well as irregular ones that may be asymmetric and/or discontinuous. The cells within a neighbourhood can also be associated varying weights for situations when spatial autocorrelation needs to be taken into account. pRPL supports multi-layer algorithms that are commonly used in geospatial applications. pRPL provides multiple data-decomposition methods for users, including row-based, column-based, and block-based decomposition, as well as a spatially-adaptive quad-tree-based (QTB) decomposition method for cases when the computational intensity is extremely heterogeneous over space. The "Update-on-Change" and Value-headed Global-index Stream techniques developed for pRPL helped to reduce the communication overhead for data exchange among the processors, hence reduce the computing time. Furthermore, the "edgesFirst" and non-blocking communication techniques overlap the computation and communication, which also helps reduce the computing time. pRPL organizes processors into groups, and supports data-task hybrid parallelism which is innovative for parallel raster processing and especially useful when handling massive-volume datasets and a large number of parallelizable tasks at the same time. With grouped processors, dynamic load-balancing can be implemented with ease. pRPL also provides an intuitive programming guideline for users to implement application-specific algorithms, and requires minimal parallel programming knowledge. Written in the C++ language and based on the Message Passing Interface (MPI) that is supported by most parallel computers, pRPL provides transparent parallelism for GIS scientists and professionals to exploit the great computing power of a large range of high-performance computing facilities, including massive super computers, computer clusters, computational grid,

cloud computing services, and multi/many-core computers. More importantly, pRPL provides a test-bed for computationally intensive geospatial analysis and models, and a problem-solving environment for previously computationally infeasible approaches.

The pRPL 1.0 was released in 2008 (for more technical details, see Guan 2008). A series of modifications have been made in version 2.0, in order to improve the flexibility and usability. Some major modifications include the following.

1. The *Cellspace* class has been re-designed and re-implemented to increase the flexibility for data I/O and accommodating various data types. Specifically, an interface to the Geospatial Data Abstraction Library (GDAL) has been added to read and write data in a large range of raster formats. Parallel I/O is not implemented in version 2.0 because the current GDAL is not multi-thread safe. Also, the value of a cell within a *Cellspace* can be retrieved to a variable and updated using a variable of a data type that is different from the *Cellspace's* data type as long as the type conversion is allowed in C++. An example is given below.

```
Cellspace mySpace;

GDALDataset *pGdalDS = (GDALDataset *)GDALOpen("my.tif",
GA_Update); // Open a GeoTIFF raster file

mySpace.initByGDAL(pGdalDS, 1); // Initialize the Cellspace using
the 1st band of the GeoTIFF file. Assume the data type is Integer

double myValue = mySpace.atAs<double>(0, 0); // Retrieve the
value (integer number) of the upper-left corner of the Cellspace,
and convert it to a double-precision floating-point number

mySpace.updateCellAs<double>(0, 0, myValue); // Update the upper-
left corner cell (integer number) using a double-precision
floating-point number

mySpace.writeGDALData(pGdalDS,1); // Write the Cellspace back to
the GeoTIFF file
```

Also, NoData is now explicitly supported by pRPL. The NoData value of a raster data file will be retrieved and loaded to the *Cellspace* when initializing the *Cellspace* using the GDAL interface. Many built-in methods of the *Cellspace* class, such as *find*, *count*, and neighborhood-based options, allow users to choose an option for ignoring the NoData value during the processing. Users are also allowed to specify the NoData value for a *Cellspace*, and this information will be writing to the raster data file when saving the *Cellspace* through the GDAL interface.

2. A *CellspaceGeoinfo* class has been added to manage the geospatial information of raster data. When initializing the *Cellspace* using the GDAL interface, the geospatial reference of the raster data will be automatically loaded to the *Cellspace*. Thus the value at a geospatial coordinate within the extent of the *Cellspace* can be retrieved and updated with ease. When writing the *Cellspace* to a GDAL-supported raster data file, the geospatial reference (if there is any) will also be written to the file.

3. A *DataManager* class has been added to provide centralized data management and to facilitate transparent data decomposition and mapping, and static/dynamic load-balancing. The *DataManager* is especially useful for multi-layer algorithms and multi-step algorithms. It is used as the table of content of a book. Each *Layer* (which may include a *Cellspace* and/or multiple *sub-Cellspaces*) has an unique ID and an unique name. Unlike in version 1.0 where users have to decompose the *Cellspace* into sub-*Cellspaces* and map them to the participating processors for each *Layer*, the *DataManager* allows the users to decompose and map all *Layers* at once and guarantees the consistency of decomposition and mapping among *Layers*. An user-defined Transition (i.e., raster-processing operation) processes the necessary layers by calling the layers' IDs or names. Multiple layers can be updated during one Transition (which is not allowed in version 1.0), and all updated layers will be automatically synchronized among all participating processors.

4. A dynamic data load-balancing mechanism has been implemented. The master processor reads the raster data into sub-*Cellspaces* and distributes the sub-Cellspaces dynamically in response to worker processors' requests, and also receives processed sub-*Cellspaces* from worker processors dynamically and writes to raster files. Such a dynamic load-balancing method is particularly useful on a heterogeneous parallel computer in which the processors have different processing speed. Ad-hoc data load-balancing is not implemented in version 2.0 yet.

5. Some methods/functions have been modified to increase the flexibility and usability. For example, the "edgeFirst" option in the *Transition* base class has been removed and all data exchanges are carried out in the "edgeFirst" way. If the Transition's "needExchange" option is set ON (which means the processing requires synchronization of the "halo" cells in sub-Cellspaces among multiple processors), the *Transition* will automatically process the "edge" cells of a sub-*Cellspace* first, and then start a non-blocking data

exchange among the participating processors while continuing processing the "interior" cells.

To evaluate the usability and performance of pRPL 2.0, a parallel Land Dynamic Model (LDM, Guan et al. in preparation) was developed. The LDM is a general-purpose land-use and land-cover change (LULCC) model based on an artificial neural network (ANN). The ANN learns the spatio-temporal pattern of land dynamics and its relationships/interactions with user-selected factors (e.g., environmental and climatic factors, socio-economic factors) through historical data, and then applies the learned pattern for simulation and forecasting. Unlike other ANN-based LULCC models such as the Land Transformation Model (LTM, Pijanowski et al. 2002), ANN-CA (Li and Yeh 2002), and ANN-Urban-CA (Guan, Wang, and Clarke 2005) where the relationships between the LULCC and driving factors stay static once the simulation starts, LDM has a self-modification process that is able to change the relationships during the simulation to enable true dynamic modeling.

The parallel LDM (pLDM) was implemented using pRPL 2.0. A set of experiments were conducted to simulate the land-use change in California, USA. The dataset include multiple raster layers representing land use, distance to city centers, distance to transportation, landscape (i.e., elevation and slope), soil characteristics, and climate characteristics, at 30-meter spatial resolution. The experiments showed that the pLDM drastically reduced the computing time to a length that is feasible for actual simulation and forecasting, and achieved fairly high speed-ups and efficiencies. Experiments also showed the pLDM scaled reasonably well as the number of processors increased and as the data size increased.

In conclusion, through a serious of modifications based on version 1.0, the flexibility and usability have been largely improved in pRPL 2.0. The data load-balancing mechanism implemented in version 2.0 helps improve the performance. Some of the key features of pRPL remain in version 2.0, such as the spatially-adaptive QTB decomposition, data-task hybrid parallelism, "edgeFirst" processing, and non-blocking data exchange. Parallel I/O and ad-hoc load-balancing have not been implemented in pRPL 2.0, but are our future plans.

References:

Guan, Qingfeng, and K. C. Clarke. In preparation. Land Dynamic Model: Enabling Dynamic Simulation and Integration for Land-use and Land-Cover Change Modeling

Guan, Qingfeng. 2008. "Getting Started with pRPL." http://www.geog.ucsb.edu/~guan/pRPL/Getting_started_with_pRPL.pdf.

Guan, Qingfeng, and K. C. Clarke. 2010. “A General-purpose Parallel Raster Processing Programming Library Test Application Using a Geographic Cellular Automata Model.” *International Journal of Geographical Information Science* 24 (5) (May): 695–722.

Guan, Qingfeng, Liming Wang, and Keith Clarke. 2005. “An Artificial-Neural-Network-based, Constrained CA Model for Simulating Urban Growth.” *Cartography and Geographic Information Science* 32 (4): 369 – 380.

Li, Xia, and Anthony G. O. Yeh. 2002. “Neural-network-based Cellular Automata for Simulating Multiple Land Use Changes Using GIS.” *International Journal of Geographical Information Science* 16 (4): 323–343.

Pijanowski, Bryan C., Daniel G. Brown, Bradley A. Shellito, and Gaurav A. Manik. 2002. “Using Neural Networks and GIS to Forecast Land Use Changes: a Land Transformation Model.” *Computers, Environment and Urban Systems* 26 (6): 553–575.