The Implementation of a Pipe-lined Grid Coverage and Grid Coverage Processor

Hong Gab Kim, Kyung Ok Kim

Spatial Information Technology Center, Computer & Software Laboratory, ETRI
161 Kajong-Dong, Yusong-Gu, Taejon 305-350, KOREA
Phone : +82-42-860-5327
Fax : +82-42-860-4844
E-mail : { gabbi, kokim }@etri.re.kr

Abstract

This paper describes the way to embody the grid coverage and the grid coverage processor which can construct pipelines. The pipeline constructed by the developed grid coverages has internal pipelines that have different resolution and it provides the way of access to very large datasets efficiently. Several operations, such as filtering, image enhancement and band operation, are embedded in the developed grid coverage and grid coverage processor COM components. The practical usefulness of the developed grid coverage and grid coverage processor has been proven by applying them in developing an image processing software for very large images.

1. Introduction

With the advent of new high resolution satellites and the amount of data significantly increasing, it is imperative that the software meets the challenges of processing such large datasets[1][2]. The Open GIS Consortium (OGC) released Grid Coverage Implementation Specification on January 2001, one of whose goals is to allow for efficient access to very large datasets up to several gigabytes in size. GP_GridCoverageProcessor, the optional interface in this specification, provides operations for different ways of accessing the grid coverage values as well as image processing functionality. This interface has been designed to allow the adaptations to be done in a pipe-lined manner. The interface does not need to make a copy of the source grid data. Instead, it can return a grid coverage object which applies the adaptations on the original grid coverage whenever a block of data is requested. In this way, a pipeline of several grid coverages can be constructed cheaply and it can treat large dataset very efficiently and fast. Unfortunately, the implementation specification released by OGC only describes "what" interfaces and methods have to be implemented and doesn't mention "how" they can be embodied. This paper explains the way to develop the grid coverage and the grid coverage processor which can treat large datasets in pipe-lined manner and its applications.

2. Grid Coverage and Grid Coverage Processor

In 1994, the Open GIS consortium (OGC) was founded in response to wide-spread recognition of the problem of non-interoperability[3]. Through the OGC standards development process, it introduced Open GIS Implementation Specification for Grid Coverage[4]. It is composed of three packages such as general coverage specification(CV) grid coverage(GC), and grid coverage processing(GP). GP_GridCoverageProcessor, the optional

interface in GP, provides operations for different ways of accessing the grid coverage values as well as image processing functionality. This interface has been designed to allow the adaptations to be done in a pipe-lined manner. The interface does not need to make a copy of the source grid data. Instead, it can return a grid coverage object which applies the adaptations on the original grid coverage whenever a block of data is requested. In this way, a pipeline of several grid coverages can be constructed cheaply and it can treat large dataset very efficiently and fast. Figure 1 illustrates how to create a grid coverage and to perform a simple image processing operation, threshold, in the pipeline.

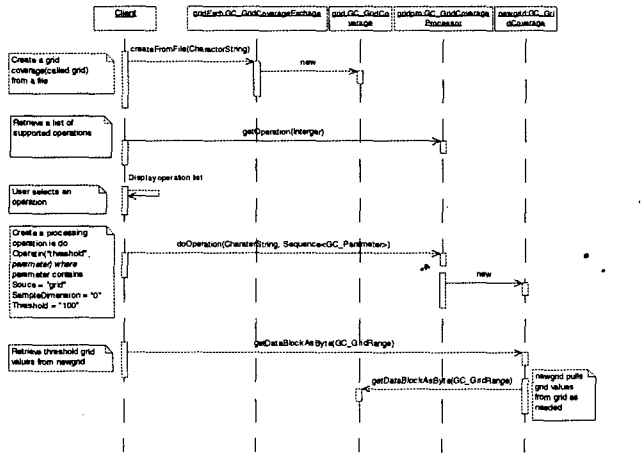

Figure 1. Grid coverage processing using the discovery mechanism.

3. Implementation

In the developed grid coverage COM component, a grid coverage processor only plays part to create new grid coverage object and add it into a pipeline. Access to datasets is performed in a grid coverage. A grid coverage processor is a factory to create a new grid coverage object and is allowed for access to member variables of the new grid coverage object. It means that a grid coverage processor is depend upon the grid coverages to be created by it. A grid coverage processor is destroyed after it completes its duty to create a new grid coverage object and add it into the pipeline.

A block of datasets flows in the pipeline only when the client requests them. On being passed to the client, the block of data is destroyed in the pipeline. Thus, the amount of data flowing in the pipeline is equal to the size of data requested by the client with getDataBlockAsXXX method regardless of the number of grid coverages in the pipeline and the total size of raster datasets. Figure 2 shows pseudo code for doOperation, a method of a grid coverage processor, and getDataBlockAsXXX, a method of a grid coverage.

```
//m_SourceGridCoverage : pointer to GC_GridCoverage interface of source grid coverage
GC_GridCoverage MyGridCoverageProcessor::doOperation
            (CharacterString opeartionName, Sequence<GC_Parameter> parameters)
//parameter[0] contains the source grid coverage interface
{
  If operationName == "Operation1" && parameters[0].name = "SourceGridCoverage"
  {
    MyGridCovearge GridCoverageObject = new MyGridCoverage
    GridCoverageObject.m_SourceGC = parameters[0].value
    //m_SourceGC is not a interface but member variable of MyGridCoverage class
    return cast<GC_GridCoverage>GridCoverageObject
  }
  Else
  {
    //no matched operation
  }
}


Sequence<XXX> MyGridCoverage::GetDataBlockAsXXX(GC_GridRange GridRange)
{
  //Request a block of raster data to the source grid coverage
  Sequence<XXX> SourceData = m_SourceGridCoverage.GetDataBlockAsXX(GridRange)
  Process( SourceData ) //perform processing fetched block of data
  Return SourceData          //return processed data
}
```

Figure 2. Pseudo code for doOperation method and getDatablockAsXXX method

```
GC_GridCoverage MyGridCoverage::getOverview(Integer overviewIndex)
{
  //m_SourceGridCoverage : pointer to GC_GridCoverage interface of source grid coverage
  MyGridCoverage OverviewObj = new MyGridCoverage //clone to create an overview
  OverviewObj.m_numOverviews = 0
  //m_numOverview is a member variable of MyGridCoverage class.
  //an overview does not have other overviews
  OverviewObj.m_SourceGridCoverage =
                mSourceGridCoverage->getOverview(overviewIndex)
  //create pipeline for overviews
  return cast<GC_GridCoverage>OverviewObj
}
```

Figure 3. Pseudo code for getOverview method

When a source gird coverage supports overviews with different resolutions, there is no need to write additional code to process data fetched from overview of the source grid coverage in a new grid coverage. Instead, when an overview of new grid coverage is referenced, the internal

pipeline for the overview is constructed. Figure 3 and figure 4 show pseudo code and procedure to construct a pipeline for overviews when an overview of new grid coverage is referenced.

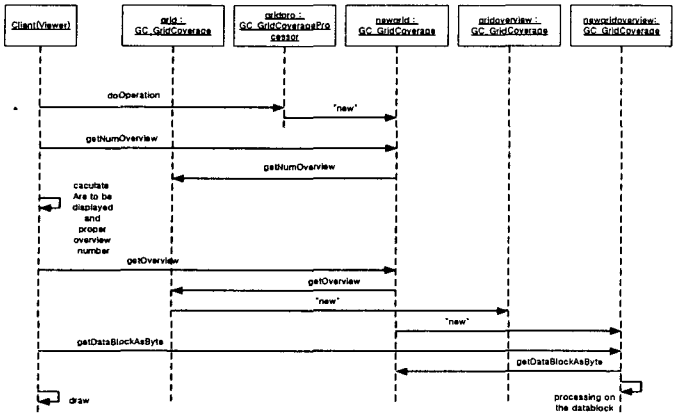

Figure 4. Internal pipeline creation

4. Application

We developed a simple image processing application to handle large image with the proposed grid coverage and grid coverage processor that can construct pipelines. This application can display very large images, the size of which is up to several gigabytes, at any zoom ratio and pan them. Also, It performs image processing, such as filtering, image enhancement and band operation, and shows immediately processed result images.

In case that a user wants to perform average filtering and band ratioing sequentially on a very large image and look over the result image on the screen, the developed application program creates a pipeline, as shown in Figure 5, with grid coverages, a grid coverage exchange and grid coverage processors by user's commands. Notice that any image processing is not performed until the result image is displayed on the screen. When the result image is about to be displayed, one of overviews is selected according to the current zoom ratio. The result block of data is fetched and displayed through the pipeline in which the selected overview is contained. Thus, the size of data block to be processed is only related with the size of display window regardless that whole image or a part of image is displayed. For example, although the size of a raster data file is up to several gigabytes, the amount of data to be loaded on memory and processed is only several megabytes when the size of a viewer is 1024 x 768.

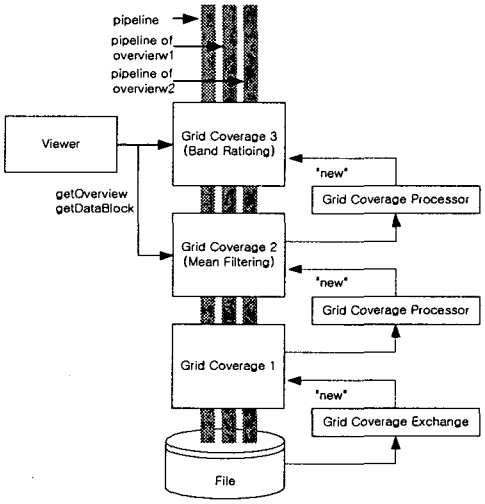

Figure 5. An example of a pipeline

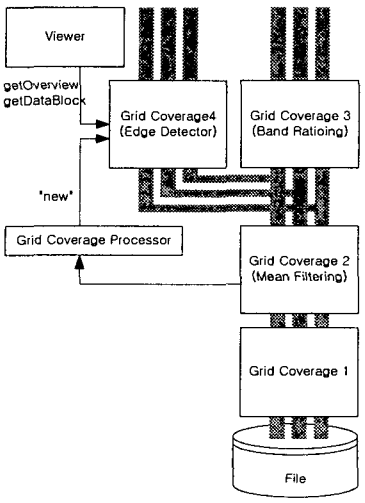

Figure 6. A branched pipeline

One of characteristics of pipeline method, that it does not create intermediate results, is widely used to develop applications for large datasets from undo/redo operation to visual programming. In Figure 5, If you want to remove

band ratioing operation from the constructed pipeline and to add edge detection operation into the pipeline, it is very simply done by creating a new edge detection grid coverage object, the source of which is grid coverage 2. The result image is fetched through the branch of the pipeline by requesting processed data to the new grid coverage, as shown in figure 6. This process of creating a branch pipeline is similar to construction of data flow model in visual programming. Thus, these grid coverages can be used for developing a visual programming tool.

5. Conclusion

The way to embody grid coverage and grid coverage processor, which can construct pipelines, has been explained. The pipeline constructed by grid coverages has internal pipelines that have different resolution and provides the way of access to very large datasets efficiently. The practical usefulness of the grid coverage and the grid coverage processor has been proven by applying them to develop an image processing software for very large images

6. References

[1] Earth Resource Website, Interactive Image Procesing using Algorithms, http://ermapper.com

[2] Owen, M.J, Lui, A.K, Lo, E.H.S, Grigg, M.W, The design and implementation of a progressive on-demand image dissemination system for very large images, *Computer Science Conference 2001, ACSC2001. Proceeding 24th Australia*, 148-155, 2001

[3] OpenGIS Consortium Technical Committee, *The OpenGIS Guide*, 1998

[4] OpenGIS Consortium, *OpenGIS Implementation Specification : Grid Coverage, Revision 1.0*, 2001