MPI-GIS: An MPI System for Big Spatial Data

Satish Puri, Marquette University
Sushil Prasad, Georgia State University

I. Introduction

Spatial Data types:

1) Vector data : co-ordinate representation.
   e.g., POLYGON ((30 10, 40 40, 20 40, 30 10))
2) Raster data : pixel format (matrix of cells).

Application Domains:

1. Geographic Information System.
2. Spatial Databases
3. Computational Geometry
4. Computer Graphics
5. VLSI CAD

Big Spatial Data Challenges using MPI:

- Buffer Count problem: MPI functions count parameter is an Integer. Character buffer size limited by 2GB and Floating point buffer size limited by 8 GB.

- MPI-IO supports unformatted data e.g., fread/fwrite. Raw Polygonal and Vector data is collection of strings which need to be parsed. MPI does not support formatted data e.g., fprintf, fscanf.

- Current parallel I/O libraries like PnetCDF and HDF5 does not support GIS vector data. Big data systems like SpatialHadoop and HadoopGIS support vector data but are not optimized to leverage HPC resources.

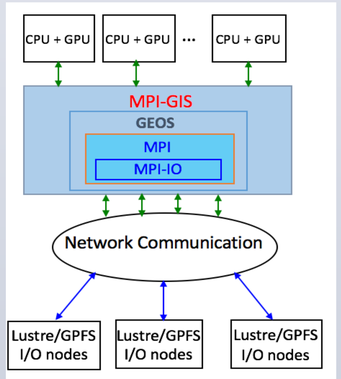

Fig.1. MPI-GIS is built on top of MPI-IO. It uses GEOS open-source library for geometric computations.

II. MPI-Vector-IO

Parallel I/O support for large vector data (e.g. OpenStreetMap) using MPI IO on GPFS parallel filesystem. Unlike PnetCDF, our implementation is not tied to any specific file format. By using a flexible interface, it allows user-defined methods to parse vertex coordinates to GEOS geometry objects.

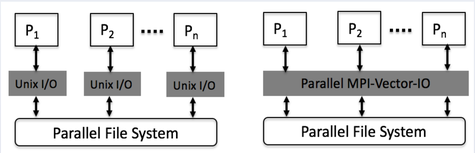

Fig. 2. Two different ways to access spatial data by parallel programs.

Fig. 3. File partitioning among processes Pi in contiguous mode.

New load balancing technique during the file partitioning phase by block-cyclic and non-contiguous file reading: Using this technique, an MPI process gets geometries from different areas which leads to load-balancing for spatial query operations during parallel I/O phase itself without explicit spatial grid based partitioning thus eliminating expensive communication phase.

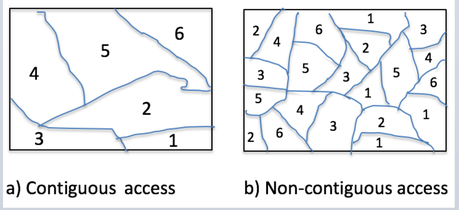

Fig. 4. Spatial partitioning as result of two different file access modes among 6 processes.

III. Spatial data aware MPI

Derived spatial data types (MPI_POINT, MPI_RECT, etc.,), spatial reduction (e.g., MPI_Reduce with MPI_UNION operator on MPI RECT type) and communication support for spatial data using MPI. With these new spatial types, the efficiency of built-in reduction operations can be leveraged.

Spatial Type (*MPI_Datatype*)	Spatial Reduction	
	Operator (*MPI_Op*)	Supported Types
MPI_POINT MPI_LINE MPI_RECT MPI_POLYGON	MPI_MIN MPI_MAX MPI_UNION	RECT, LINE, POINT RECT, LINE, POINT RECT

MPI-GIS Example

Spatial type: Rect *in_rect, *out_rect; Line *buffer;
MPI-GIS operator: **MPI_UNION**

IO :
MPI_File_read_all(file, buffer, 200, **MPI_LINE**, status);

Reduction:
MPI_Reduce(in_rect, out_rect, 100, **MPI_RECT**, **MPI_UNION**, 0, MPI_COMM_WORLD);

Communication:
MPI_Alltoall(in_rect, 100, **MPI_RECT**, out_rect, 100, **MPI_RECT** , MPI_COMM_WORLD);

Fig. 5. MPI-GIS spatial data types embedded within MPI built-in file I/O, communication and reduction functions.

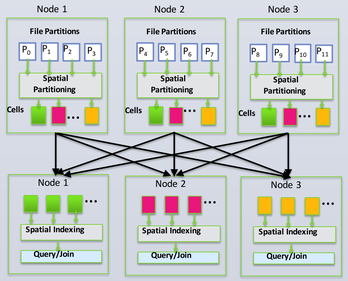

Fig. 6 Data Flow showing different phases for spatial partitioning, indexing for Spatial Query and Join with Filter and Refine subsystems. MBRs are used for filtering out undesirable polygon pairs and polygon vertices are used for refinement phase.

IV. Experimental Results

Roger Supercomputer at UIUC/NCSA
Each node with two Intel Xeon E5 (2.6GHz, 10 cores each).
Each node has 256GB of physical RAM.
IBM General Parallel File System (GPFS).
MPICH/3.1.4 and GCC 4.9.2.
Geometry Engine - Open Source (GEOS) library version 3.4.2

	Dataset	Shape	File Size	Count	Seq. I/O+Parsing (sec)	Parallel I/O+Parsing (sec)
1	Cemetery	Polygon	56 *MB*	193K	2.1	0.27, p = 20
2	Lakes	Polygon	9 *GB*	8*M*	328	14, p = 45
3	Roads	Polygon	24 *GB*	72*M*	786	23, p = 75
4	All Objects	Polygon	92 *GB*	263*M*	4728	76, p = 90
5	Road Network	Line	137 *GB*	717*M*	2873	62, p = 160
6	All Nodes	Point	96 *GB*	2.7*B*	3782	112, p=160

Table 1: Sequential and Parallel I/O + Parsing time for real-world datasets.

Fig. 7. Sequential and Parallel I/O time comparison using datasets of different size and types.

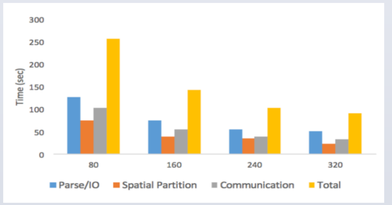

Fig. 8. Execution time breakdown for spatial data partitioning of 137 GB Road network data with 717 million geometries partitioned among 2048 grid cells for different number of MPI processes.

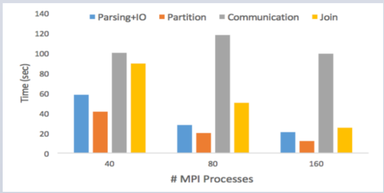

Fig. 9. Execution time breakdown for spatial join of Lakes data (9 GB, 8 Million polygons) with Cemetery data (56 MB, 190K polygons) using different number of MPI processes.

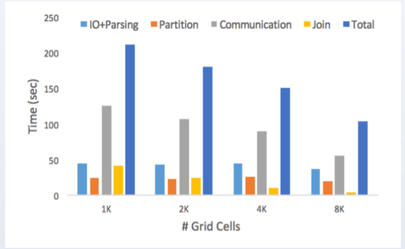

Fig. 10. Execution time breakdown for different number of grid cells for Spatial Join (Lakes, Cemetery) using 80 MPI processes.

V. Future Work

Load-balanced GIS algorithms using MPI and CUDA on CPU-GPU cluster.

VI. References

1. Danial A, S. Puri, S. Prasad, GCMF: An Efficient End-to-End Spatial Join System over Large Polygonal Datasets on GPGPU Platform, SIGSPATIAL, 2016
2. S. Puri and S. Prasad, *Output-Sensitive Parallel Algorithm for polygon clipping*, International Conference on Parallel Processing (ICPP), 2014.
3. S. Puri and S. Prasad, *MapReduce algorithms for GIS polygonal overlay processing*, IEEE International Parallel and Distributed Processing Symposium Workshops , 2013.
4. S. Puri and S. Prasad,A Parallel Algorithm for Polygon Clipping with Improved Bounds and a Distributed Overlay Processing System using MPI, CCGrid, 2015.
5. D. Agarwal, S. Puri, and S. Prasad, *A system for GIS polygonal overlay computation on linux cluster - an experience and performance report*, in IEEE International Parallel and Distributed Processing Symposium workshops, 2012.
6. *A High Performance Spatial Join on GPU & Parallel I/O for Vector Data*, NSF Workshop on Geospatial Data Science in the Era of Big Data and CyberGIS (held in conjunction with CyberGIS-16), Urbana Champaign, 2016 (Position Paper).