MPI-GIS: An MPI System for Big Spatial Data

Satish Puri
Mathematics, Statistics and Computer Science
Marquette University
satish.puri@marquette.edu

Sushil K. Prasad
Computer Science Department
Georgia State University
sprasad@gsu.edu

ABSTRACT

In recent times, geospatial datasets are growing in terms of size, complexity and heterogeneity [6]. High performance systems are needed to analyze such data to produce actionable insights in an efficient manner. For polygonal a.k.a vector datasets, operations such as I/O, data partitioning, and communication becomes challenging in a cluster environment. In this work, we present *MPI-GIS* equipped with *MPI-Vector-IO*, a parallel I/O library that we have designed using MPI-IO specifically for irregular vector data formats such as *Well Known Text*, *CSV*, etc. Our system can perform spatial in-memory indexing and join efficiently for an order of magnitude larger datasets compared to our previous work. It makes MPI aware of spatial data and spatial primitives and provides support for spatial data types embedded within collective computation and communication using MPI message-passing library.

With *MPI-Vector-IO*, it takes less than 2 minutes to scan through 2.7 billion geometries in 96 GB file using 160 MPI processes as opposed to about an hour sequentially on GPFS parallel filesystem. For unformatted binary data, it takes about 2 seconds to scan through 700M edges in 11 GB file using 200 processes. In general, the I/O and parsing are improved by one to two order of magnitude over real-world datasets.

Keywords

GIS; Polygon Overlay; MPI IO; GPU; CUDA

1. INTRODUCTION

With the advent of big spatial data, there is an increasing demand for efficient geo-spatial algorithms for indexing and querying that can scale with the input data size and the size of HPC cluster [6]. This paper talks about challenges encountered in handling spatial big data using MPI, injecting spatial data awareness in MPI and parallel I/O using MPI-IO while extending MPI-GIS [7]. Much of the research on big spatial data has been done on top of HDFS filesystem using MapReduce paradigm [4]. With time-critical and latency sensitive applications in mind, we are proposing HPC-oriented solution to geo-spatial data problems based on MPI.

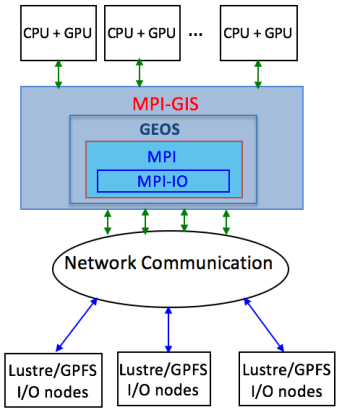

Figure 1: Design of *MPI-GIS* on a parallel I/O architecture.

Our system uses *Geometry Engine OpenSource (GEOS) library* that provides 1) spatial data structures including Quadtree and R-tree, 2) computational geometry and GIS algorithms, and 3) parsing geometries. By partitioning the file and spatial data, MPI-GIS enables the usage of GEOS in a distributed MPI environment. Figure 1 shows the system architecture.

Supporting parallel I/O for polygonal data (vector): *Well-Known Text (WKT)* is a text markup language for representing vector geometry objects on a map. A polygon with 3 vertices is represented as POLYGON ((30 10, 40 40, 20 40, 30 10)). XML is also used to represent co-ordinates of polygons and other shapes. MPI only provides functions for unformatted binary file I/O similar to fread/fwrite in C language. It does not provide functions for formatted text based spatial data (WKT, CSV, XML) which is the case with the vector data. MPI-Vector-IO component is designed to address this issue.

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

Figure 2: MPI derived spatial data types and reduction operators.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Example usage

```
Spatial type: Rect *in_rect, *out_rect;
      MPI-GIS operator: MPI_UNION
1)    MPI_Allreduce(in_rect, out_rect, 1, MPI_RECT, MPI_UNION, MPI_COMM_WORLD);
2)    MPI_Bcast(in_rect, 1, MPI_RECT, 0, MPI_COMM_WORLD);
```

Figure 3: Example of using new types and operators in MPI reduction and communication functions.

Spatial data aware MPI: Our system is built using new derived spatial data types (MPI_POINT, MPI_RECT, etc.,), spatial reduction operations (e.g., MPI_Reduce with MPI_UNION operator on MPI_RECT type) and communication support for spatial data using MPI message-passing library. With these new spatial types, the efficiency of built-in reduction operations can be leveraged. Figure 2 and Figure 3 show a few examples.

MPI Buffer Count limitation: One of the limitations of MPI for large data is that the *count* parameter in all MPI communication and I/O functions is defined as an *integer* data type. Due to this limitation, MPI communication and I/O buffer cannot be more than 2 GB in case of character data [5]. It should be noted that the restriction is not on the size of the buffer but on the count of the number of elements. Similarly, file offsets are integer arguments in MPI I/O functions which results in an upper bound on the number of bytes that can be read by an MPI process. Our system takes care of these issues.

Collective Reduction Operators for Spatial types With derived data types, existing MPI reduction operators like MPI_MIN do not work. So, we have redefined them for lines, MBRs, etc. The *min* operator can be used to find the line/rectangle with minimum size among processes. New MPI_UNION operator on MBRs is also defined which has been used to find the grid dimensions from the union of MBRs generated by individual processes during spatial partitioning. To implement it, there is a user-defined function linked to the *union* operator that performs geometric union of rectangles. MPI uses this operator to carry out the function in an optimized reduction tree fashion. An example usage is shown in Figure 3. These operators can be non-commutative, but must be associative. Similarly, MPI_SCAN could also be used with these data types.

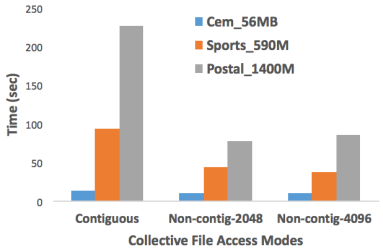

Figure 4: Spatial Join on MBRs of Lakes (500 MB) and polygonal datasets (Cemetery (56MB), Sports (590MB) and Postal (1.4GB)) using 80 MPI processes (2k and 4k block sizes). Experiment run on 4 computer nodes (20 cores per node) of ROGER Supercomputer [1] with GPFS filesystem.

MPI-IO based load-balancing: MPI provides functions for contiguous as well as non-contiguous file reading. In the latter case, each process has a different file view and reads file blocks (user specified) in a round-robin fashion. Compared to contiguous file access, this results in processes reading data from different areas in the map. Accessing geometries in a round-robin fashion can potentially aid in load-balancing. As shown in Figure 4, non-contiguous file reading with 2k and 4k block size leads to improved performance when joining the data sets in comparison to contiguous file access.

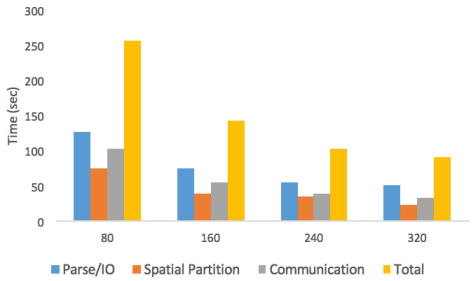

Figure 5: Execution time breakdown for 137 GB Road network data with 717 M geometries partitioned among 2048 grid cells for different number of MPI processes.

The real data used in experiments are extracted from OpenStreetMap [2]. Figure 5 below shows the breakdown of overall execution time with different number of MPI processes for global spatial data partitioning. Spatial partition phase shows the time taken for local data partitioning among grid cells. Future work includes integration with GPU/CUDA kernels for spatial join that we have already developed [3].

2. REFERENCES

[1] NCSA ROGER Cluster. https://wiki.ncsa.illinois.edu/display/ROGER/ROGER+Technical+Summary.

[2] OpenstreetMap Datasets. http://spatialhadoop.cs.umn.edu/datasets.html.

[3] D. Aghajarian, S. Puri, and S. K. Prasad. GCMF: an efficient end-to-end spatial join system over large polygonal datasets on GPGPU platform. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2016*, pages 18:1–18:10, 2016.

[4] A. Eldawy and M. F. Mokbel. Spatialhadoop: A MapReduce framework for spatial data. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 1352–1363. IEEE, 2015.

[5] W. Gropp, T. Hoefler, R. Thakur, and E. Lusk. *Using advanced MPI: Modern features of the message-passing interface*. MIT Press, 2014.

[6] S. Prasad, D. Aghajarian, M. McDermott, D. Shah, M. Mokbel, S. Puri, S. Rey, S. Shekhar, Y. Xe, R. Vatsavai, F. Wang, Y. Liang, H. Vo, and S. Wang. Parallel Processing Over Spatial-Temporal Datasets From Geo, Bio, Climate And Social Science Communities: A Research Roadmap. *6th IEEE International Congress on Big Data, Hawaii*, 2017.

[7] S. Puri and S. K. Prasad. A parallel algorithm for clipping polygons with improved bounds and a distributed overlay processing system using MPI. In *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pages 576–585. IEEE, 2015.