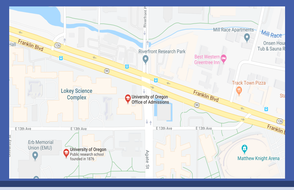
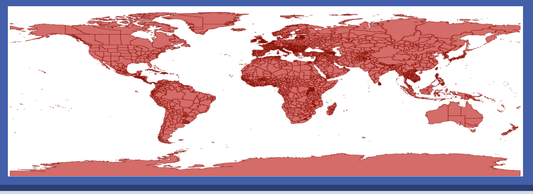


Introducing PDC concepts with spatial computing

Satish Puri, Marquette University



Spatial Computing

1. 2D co-ordinates data
   e.g. points, lines, polygons
   POLYGON ((30 10, 40 40, 20 40, 30 10))
2. Objects with location
   e.g. mobile and online maps
3. Searching geometries, finding hotspots

Objective

Development of course materials that are at the intersection of spatial computing and PDC.

Motivation

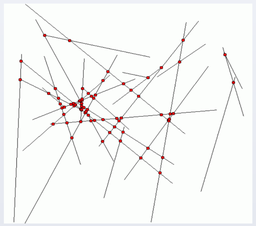
1. There is more emphasis on image data w.r.t. to PDC contents. Less emphasis on coordinate data.
2. Three day Spatial Analytics workshop at Marquette.
3. GIS Book of Knowledge interested.
4. Easy to visualize spatial data on maps.
5. Coordinate data is challenging for parallelization
   Challenges: Irregular Access Pattern, Load-imbalance due to density variation
   - Uniform, Skewed distribution

Target Audience

1. Algorithms
2. Computational Geometry
3. Computer Graphics
4. Spatial Databases

OpenACC Directives for Accelerators

OpenMP



Algorithm 1 Naive Brute Force

```
1: Load all lines to L
2: for each line l1 in L do
3:     for each line l2 in L do
4:         Test for intersection between l1 and l2
5:         if intersections exists then
6:             calculate intersection point
7:             store it in results
8:         end if
9:     end for
10: end for
```

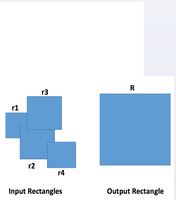
1. Loop parallelization

   on MultiCore and Manycore

```
#pragma acc parallel for
 for(int i = 0; i< objects; i++)
 {
       for(int j = 0; j< objects; j++)
       {
            intersect(line[i], line[j]);
       }
 }
```

MPI

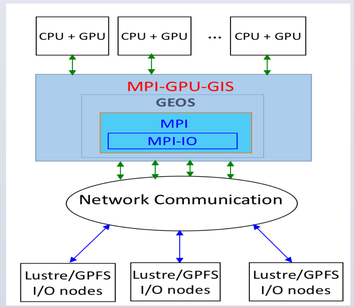
Spatial Type (*MPI_Datatype*)	Spatial Reduction	
	Operator (*MPI_Op*)	Supported Types
MPI_POINT	MPI_MIN	RECT, LINE, POINT
MPI_LINE	MPI_MAX	RECT, LINE, POINT
MPI_RECT	MPI_UNION	RECT
MPI_POLYGON		

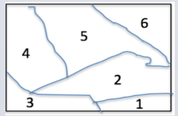


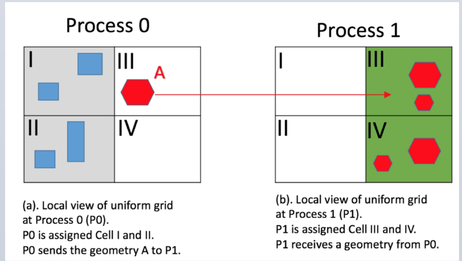
MPI-GIS Example

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

- MPI_Reduce(in_rect, out_rect, 100, MPI_RECT, MPI_UNION, 0, MPI_COMM_WORLD);
- MPI_Alltoall(in_rect, 100, MPI_RECT, out_rect, 100, MPI_RECT , MPI_COMM_WORLD);

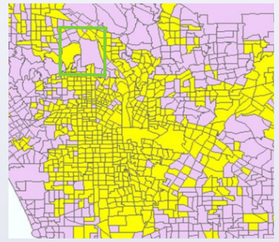






(a). Local view of uniform grid at Process 0 (P0).
P0 is assigned Cell I and II.
P0 sends the geometry A to P1.

(b). Local view of uniform grid at Process 1 (P1).
P1 is assigned Cell III and IV.
P1 receives a geometry from P0.



Range Query

Example: Rectangle query using MPI

Input: Base layer of rectangles and a given query rectangle.

Output: All rectangles from the base layer overlapping with the query rectangle should be returned.

Each MPI process reads a subset of the input polygons
Integer variables: startIndex, endIndex, numPolygons, processId, numProcs;

List of overlapping rectangles -> overlappingRectList
Steps

1. Compute startIndex, endIndex for each MPI process.

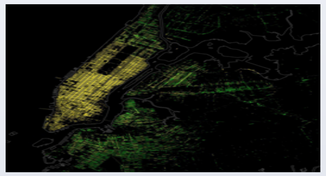
2. Each process computes Geometric Intersection (input rectangle, query rectangle).

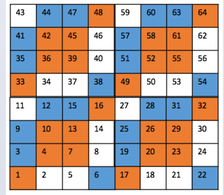
```
for ( int i = startIndex; i<endIndex; i++)
{
    if(intersect(R_i, q))
        overlappingRectList.append(R_i);

}
```

3. Process 0 aggregates the result from all processes
   MPI_Gatherv(overlappingRectList, list.size());

MapReduce

- Billion trips - New York Taxi Trip Datasets
- About a 1 TB of OpenStreet Map data





MapReduce based spatial computation

Map Phase

```
map(Geometry geom)
{
  calculate grid cellid
  emit(cellId, geom)
}
```

Reduce Phase

```
reduce(int cellId, List<Geometry> geoms)
{
  for(Geometry g: geoms)
    if(g.intersects(queryRectangle)
      emit(g);
}
```