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

Spatial Computing
1. 2D co-ordinates data
e.g. points, lines, polygons
2. Objects with location
e.g. mobile and online maps
3. Searching geometries, finding hotspots

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

Algorithm 1: Naive Brute Force
1. Load all lines to L
2. For each line l_i in L do
3. For each line l_j in L do
4. Test for intersection between l_i and l_j
5. If intersections exist 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]);
    }
}

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 needs a subset of the input polygons

Parallelization using OpenMP

Parallelization using OpenACC

Parallelization using OpenMP-ACC

Parallelization using OpenACC-ACC

MapReduce based spatial computation
Map Phase
map(Geometry geom)
{
    calculate grid cellid
    emit(cellid, geom)
}

Reduce Phase
reduce(cellid, List<Geometry> geoms)
{
    for (Geometry g: geoms)
    {
        if (g.intersects(queryRectangle))
        {
            emit(g);
        }
    }
}