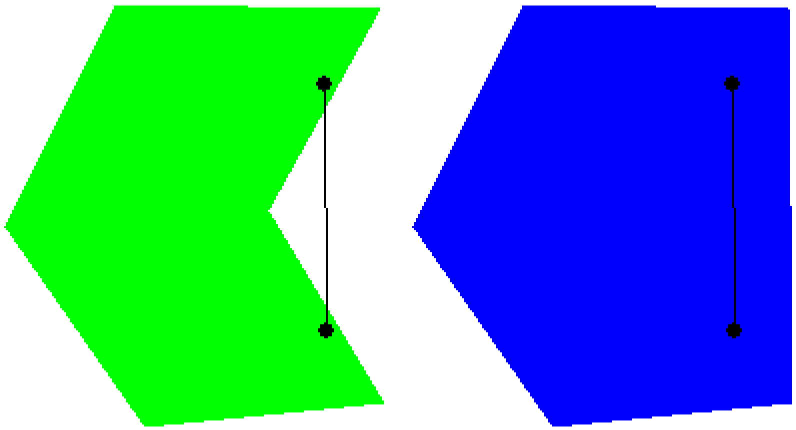
Solving Convex Hull Problem in Parallel

Anil Kumar
Ahmed Shayer Andalib
CSE 633 Spring 2014

Convex Hull: Formal Definition

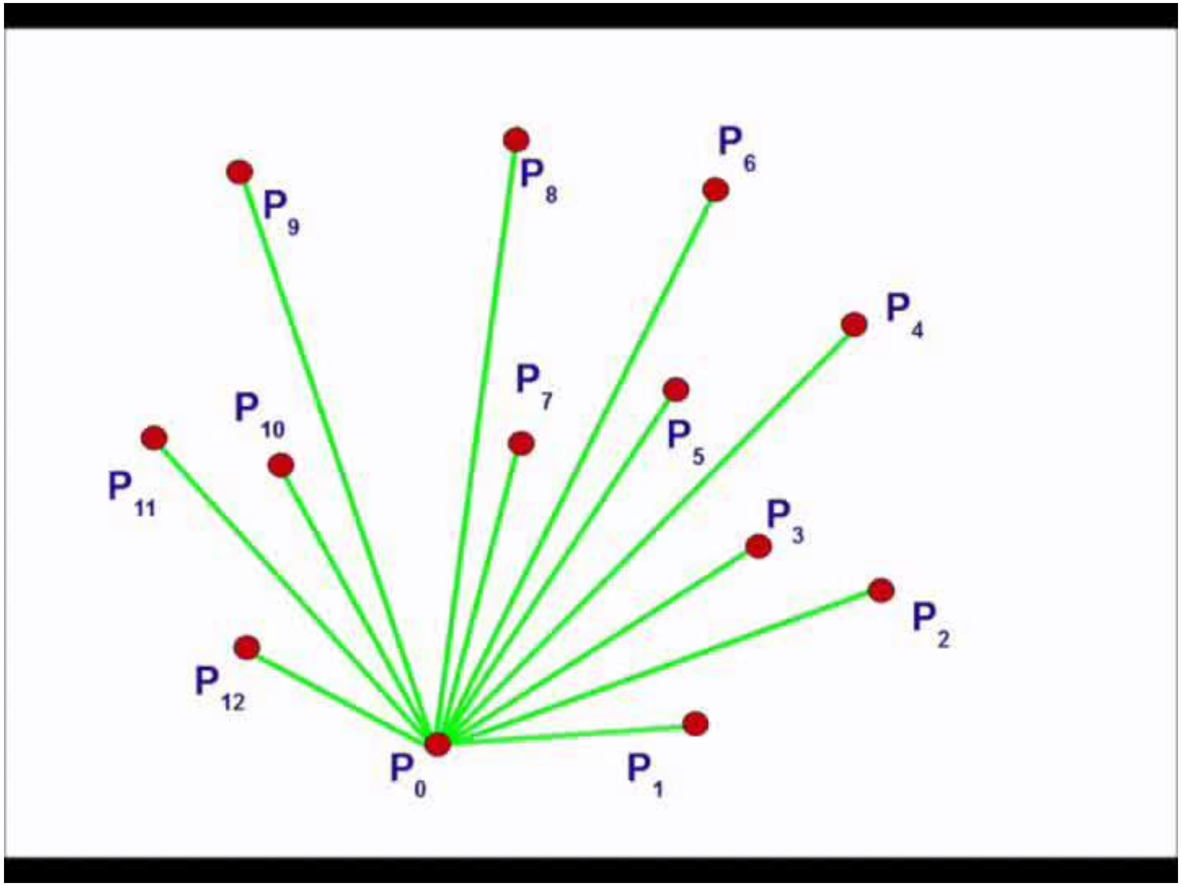
- A set of planar points *S* is convex if and only if for every pair of points x, y ∈ *S*, the line segment xy is contained in *S*.
  - Let *S* be a set of n points in the plane.
- The convex hull of *S* is defined to be the smallest convex polygon P containing all n points of *S*.



Solving The Convex Hull Problem

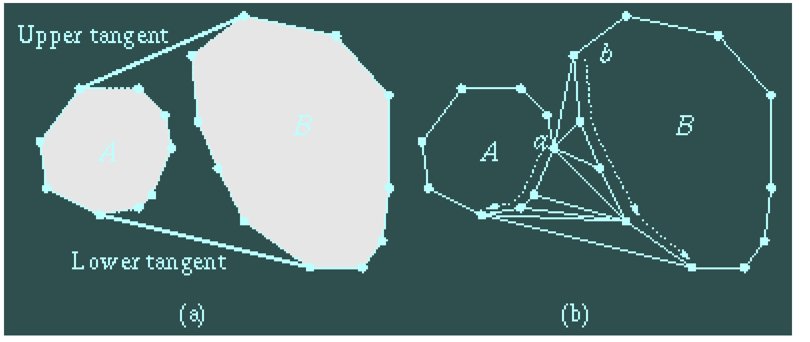
- A solution to the convex hull problem consists of determining an ordered list of points of *S* that define the boundary of the convex hull of *S*.
- This ordered list of points is referred to as hull(*S*). Each point in hull(*S*) is called an extreme point of the convex hull
- A pair of adjacent extreme points is referred to as an edge of the convex hull
- We have implemented our algorithm for solving Convex Hull in two dimensions.

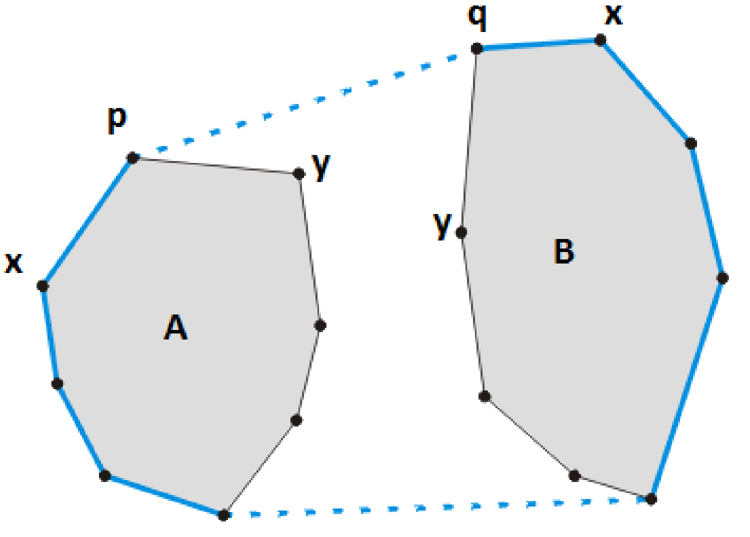
Graham’s Scan



Divide and Conquer

- Let S be the input list of points
- Partition the point set S into two sets A and B, where A consists of half the points with the lowest x coordinates and B consists of half of the points with the highest x coordinates.
- Recursively compute Convex Hull
  HA= Hull(A) and HB = Hull(B).
- Merge HA and HB: find the two edges (the upper and lower common tangents).
  - The upper common tangent can be found in linear time by scanning around the left hull in a clockwise direction and around the right hull in an anti-clockwise direction
  - Similarly determine lower common tangent
- The two tangents divide each hull into two pieces. The edges belonging to one of these pieces must be deleted.





Parallel Implementation – 1

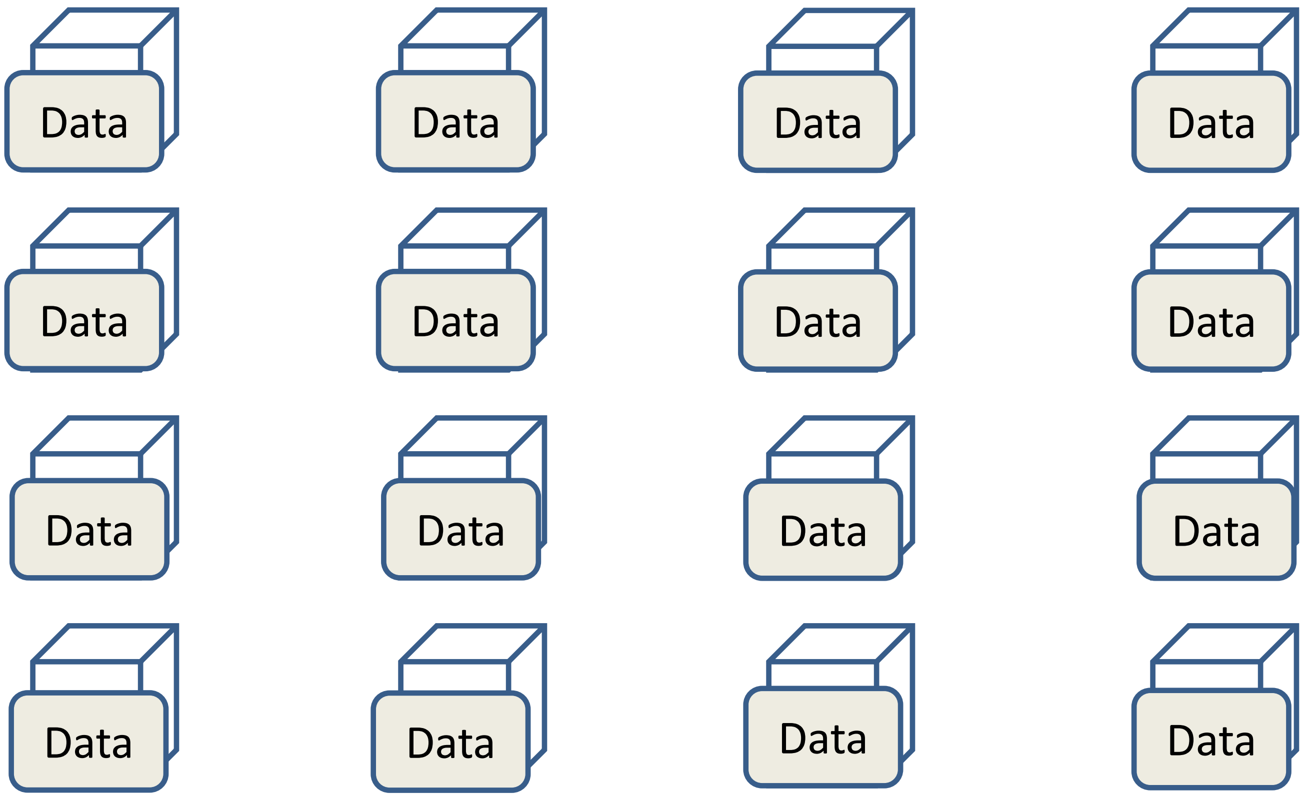
- Algorithm – Divide and Conquer
- Architecture – Mesh of size n^2 (n x n)
- Implementation:
  1. Data Generation
  2. Determination of local convex hulls
  3. Merging the convex hulls

Parallel Implementation – 1

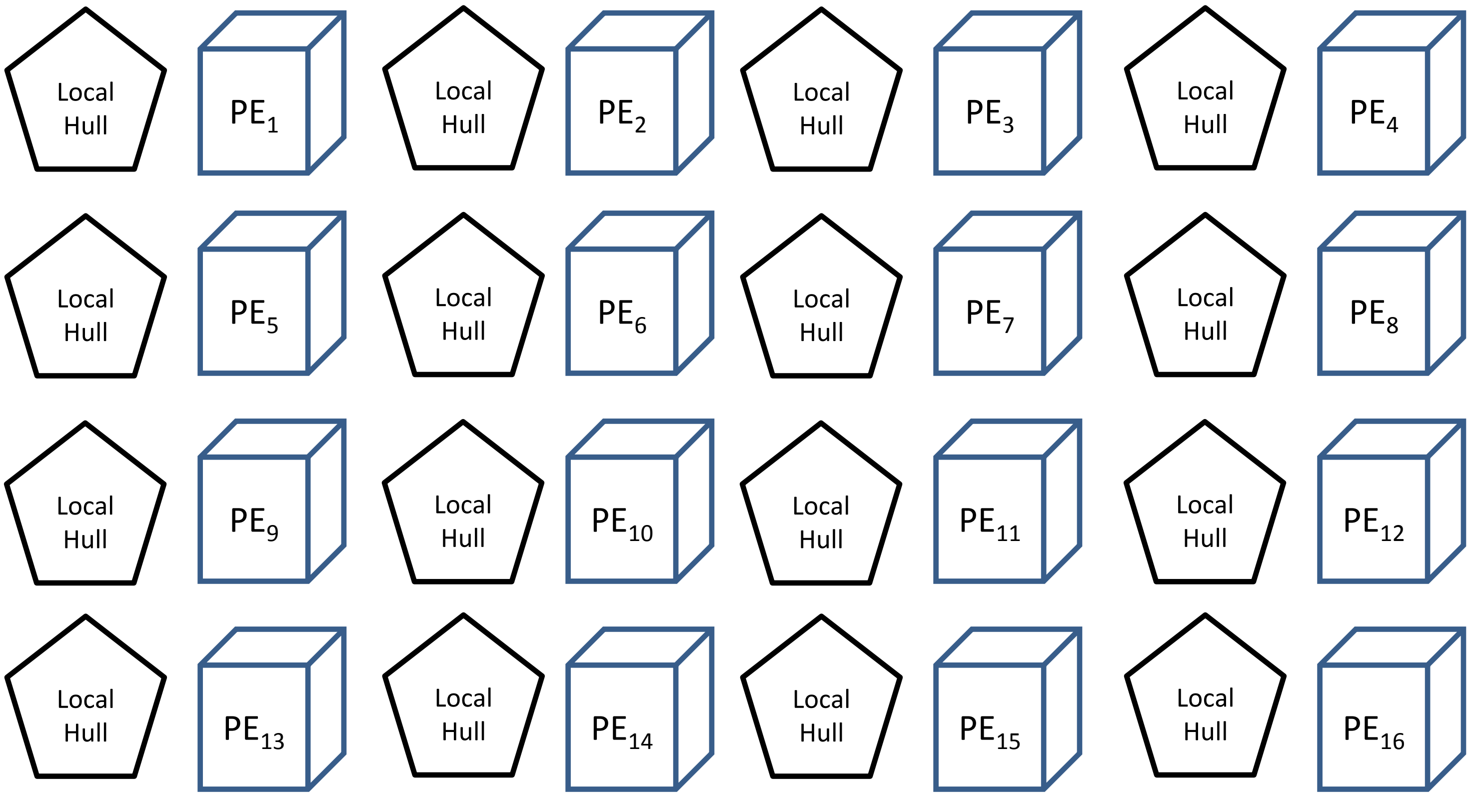
- Data generation:
  - Each processing element (PE) will generate a fixed set of point within a range of x – coordinates based on their assigned ranks
    - Ensures that data is initially partitioned based on x – coordinates
- Local Convex Hulls:
  - Each PE will compute it's local convex hull using sequential divide and conquer algorithm
- Merging the Local Convex Hulls:
  - Perform a left – to – right row – based merge operation to merge the hulls
  - The rightmost column will perform a top – down column – based merge operation to merge the hulls. Final Convex Hull will reside in the bottom PE in the rightmost column

SIMULATION: MESH OF SIZE 16 (4 X 4)

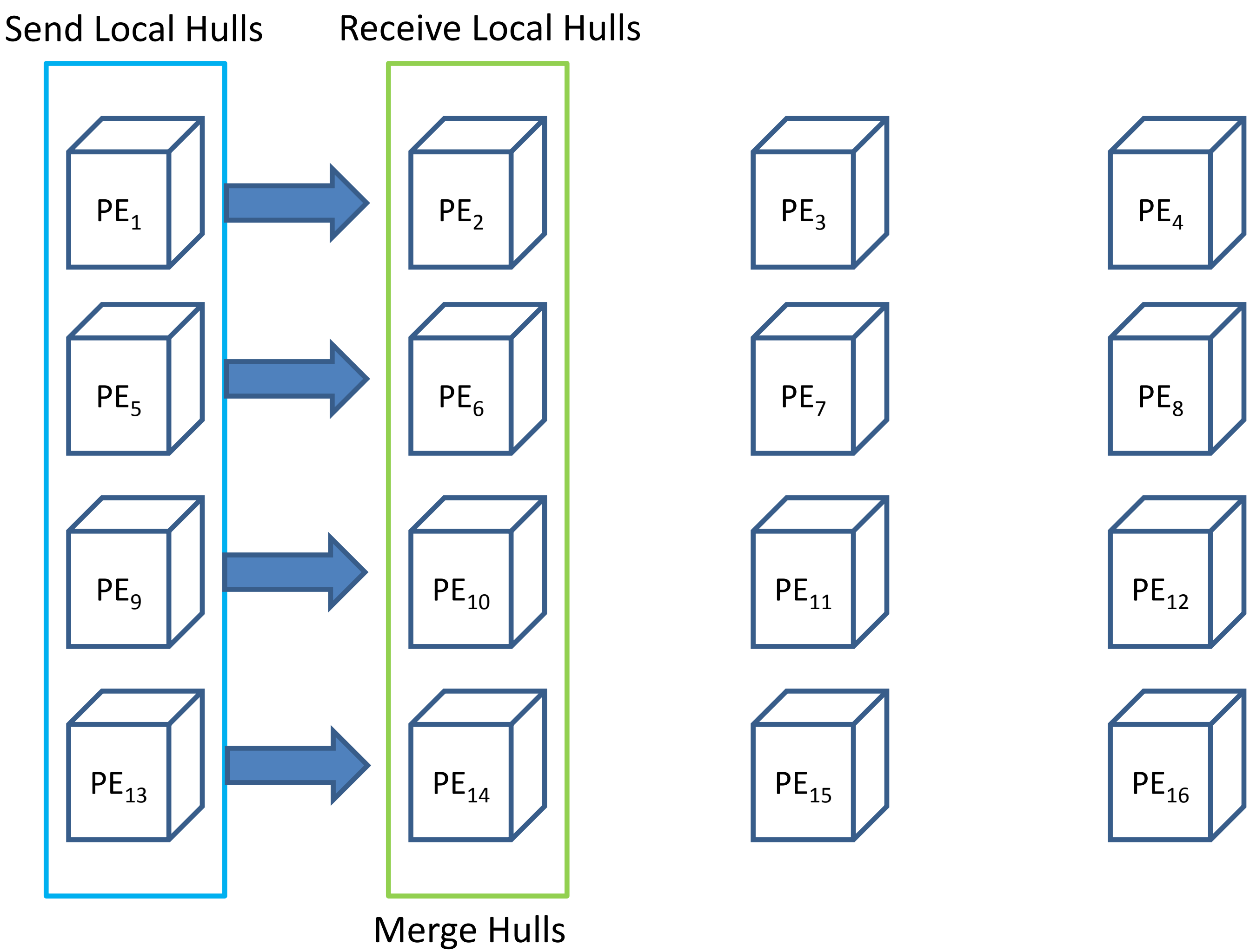
Data Generation



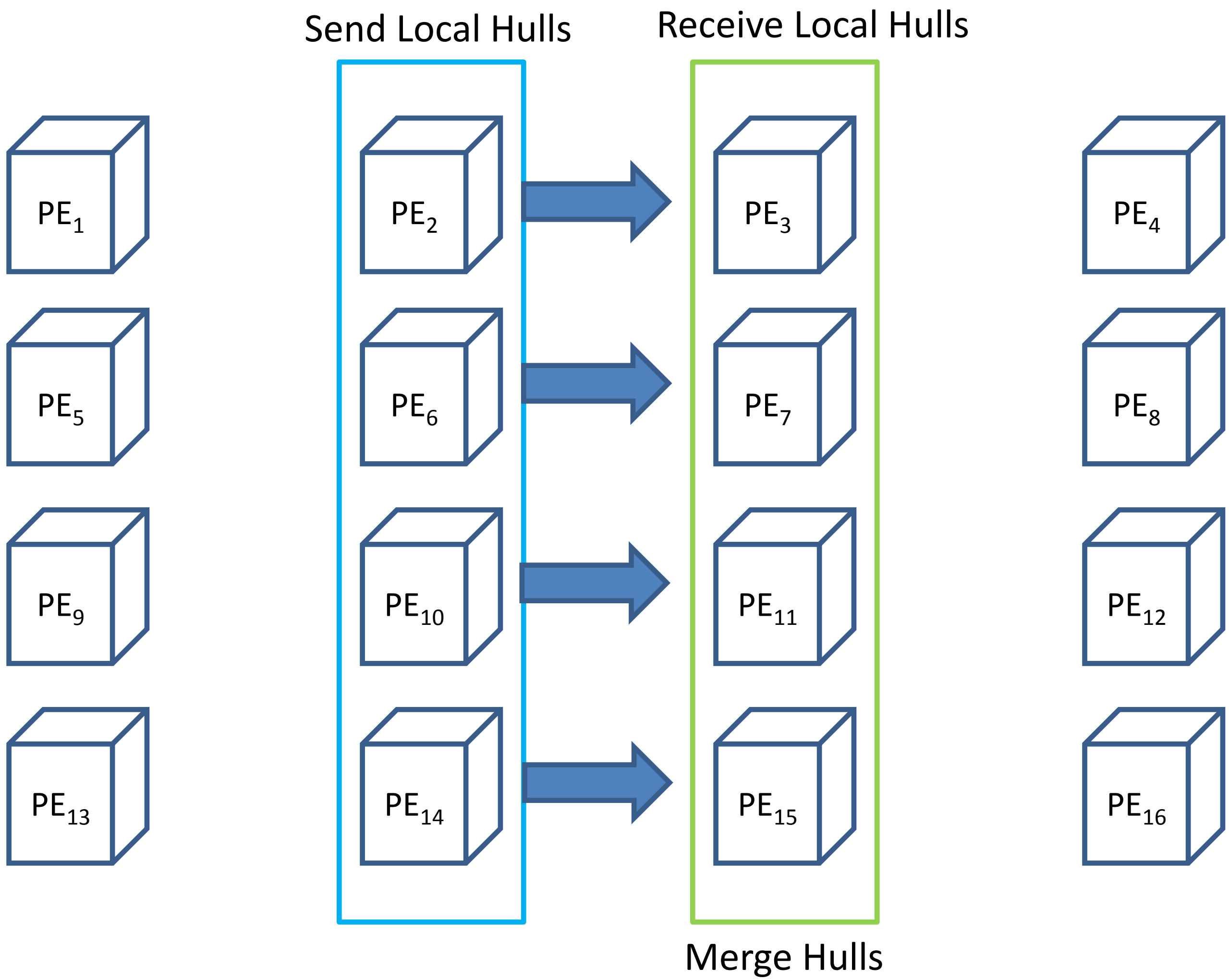
Compute Local Hulls: Sequential Divide and Conquer



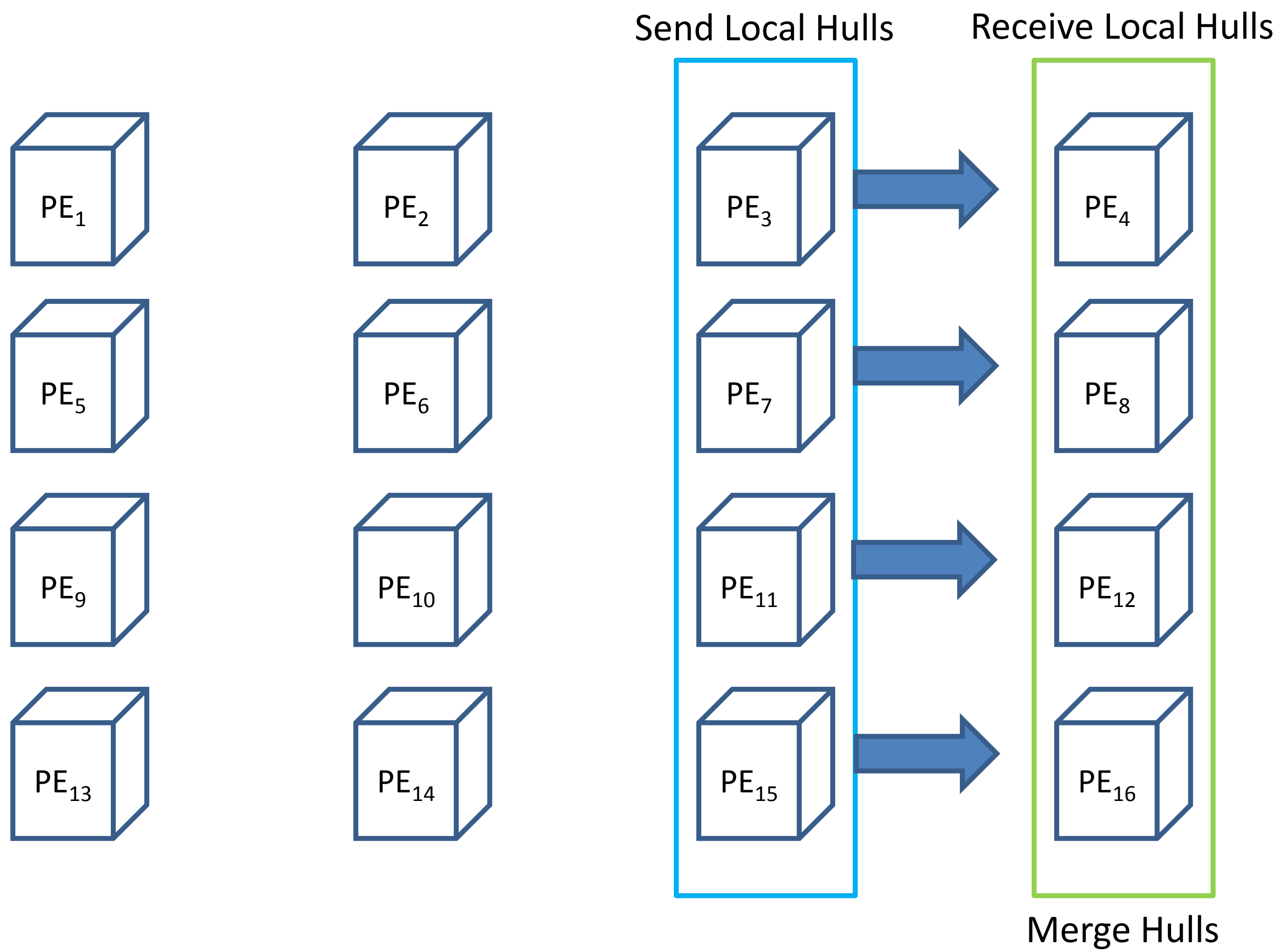
Merging Local Hulls



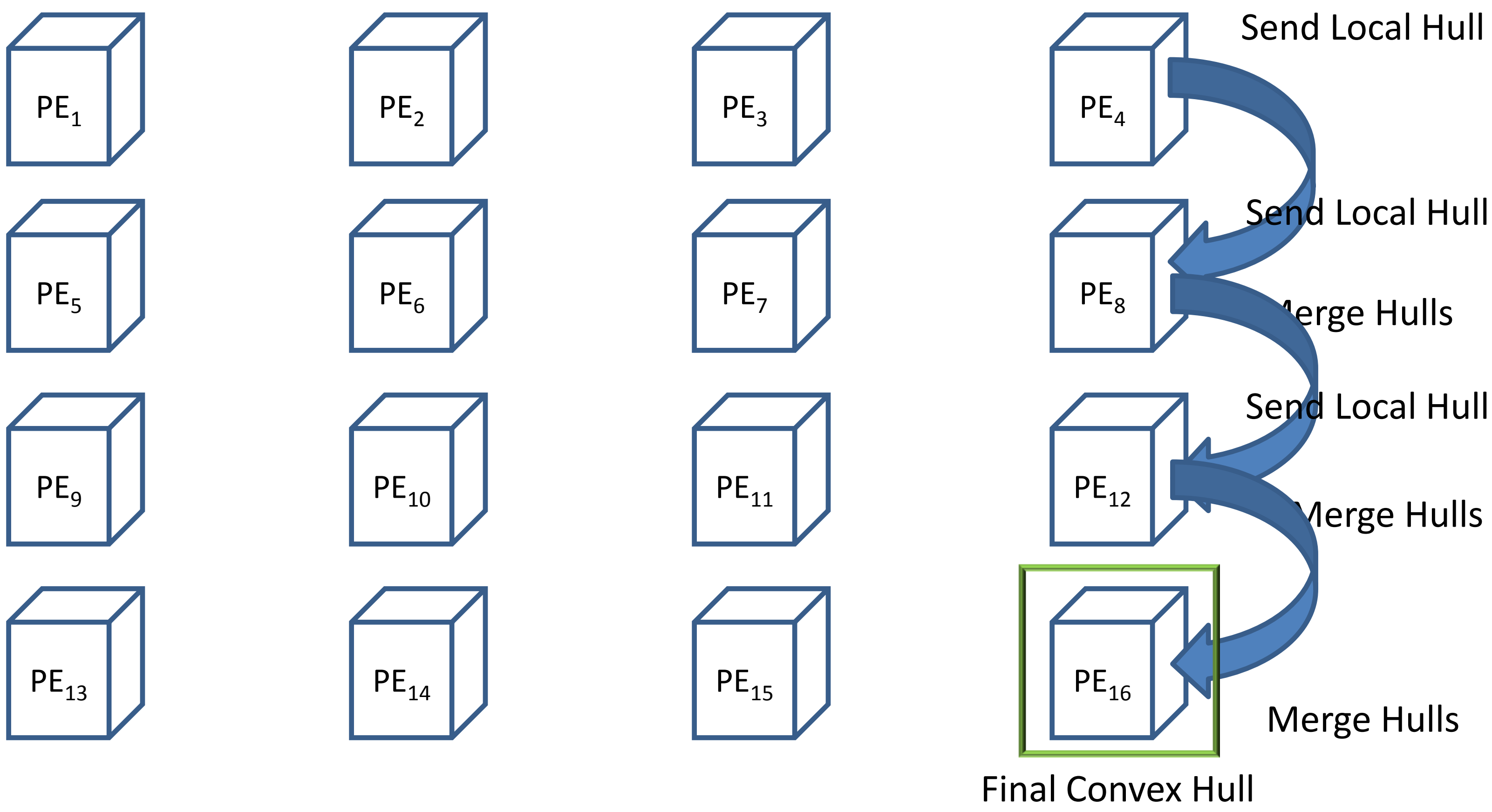
Merging Local Hulls



Merging Local Hulls



Merging Local Hulls



Mesh pseudo code

- if(col==0)
- {
  - MPI_Send
- }
- else if(col==r-1)
- {
- MPI_Recv
  - get_hull(inp,out,col);
- if(row==0)
- {
  - » MPI_Send

  }
- else{
- MPI_Recv
  get_hull(inp,out,col);
  // except the last processor MPI SEND
- MPI_Send
- }
- }
  else
- {
- MPI_Recv
  - get_hull(inp,out,col);
  - MPI_Send

  }

Parallel Implementation – 2

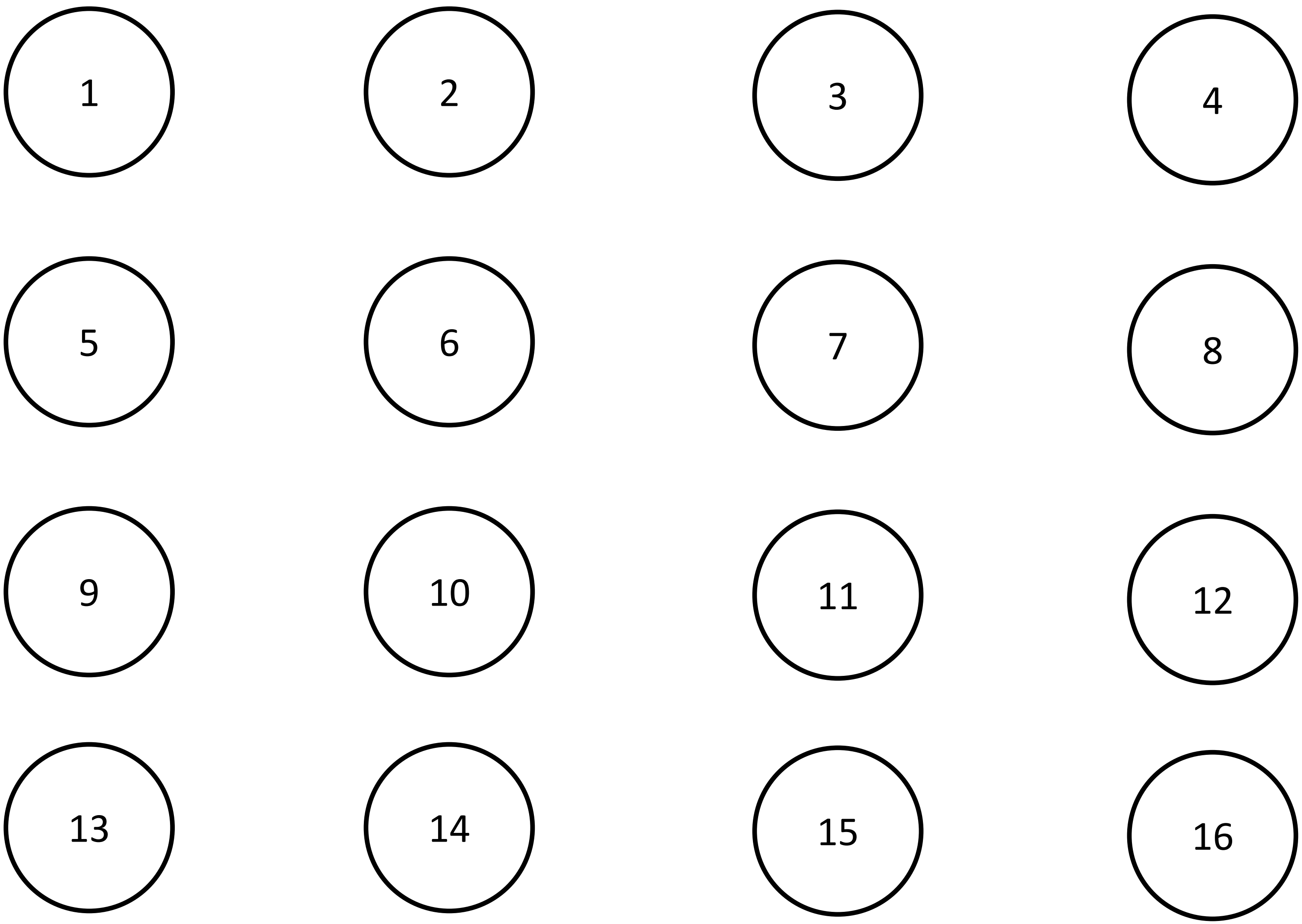
- Algorithm – Divide and Conquer
- Implementation:
  1. Data Generation
  2. Determination of local convex hulls
  3. Merging the convex hulls

Parallel Implementation – 2

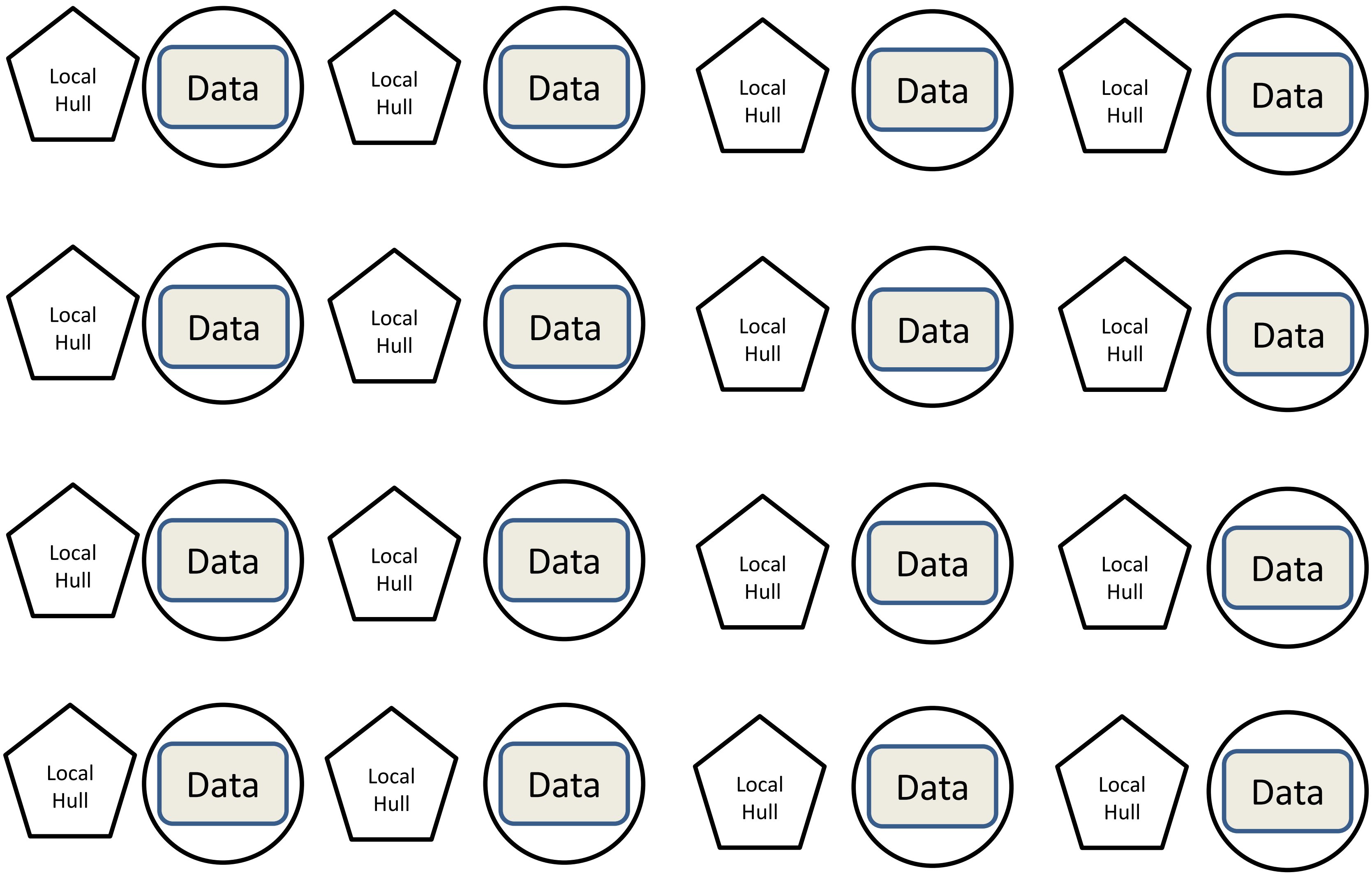
- Each PE is assigned a logical rank alongside it’s global rank
- Data Generation: Same as before
- After each PE has computed local hulls sequentially:
  1. Each PE will send it’s local hull to the PE that is next to it in logical ranking
  2. The PE that receives the hull performs merge operation
  3. The logical ranks are then updated as:
     - $rank_{logical} = rank_{logical} / 2$
- Repeat steps 1 – 3 until the final Hull is computed

SIMULATION: 16 PROCESSORS

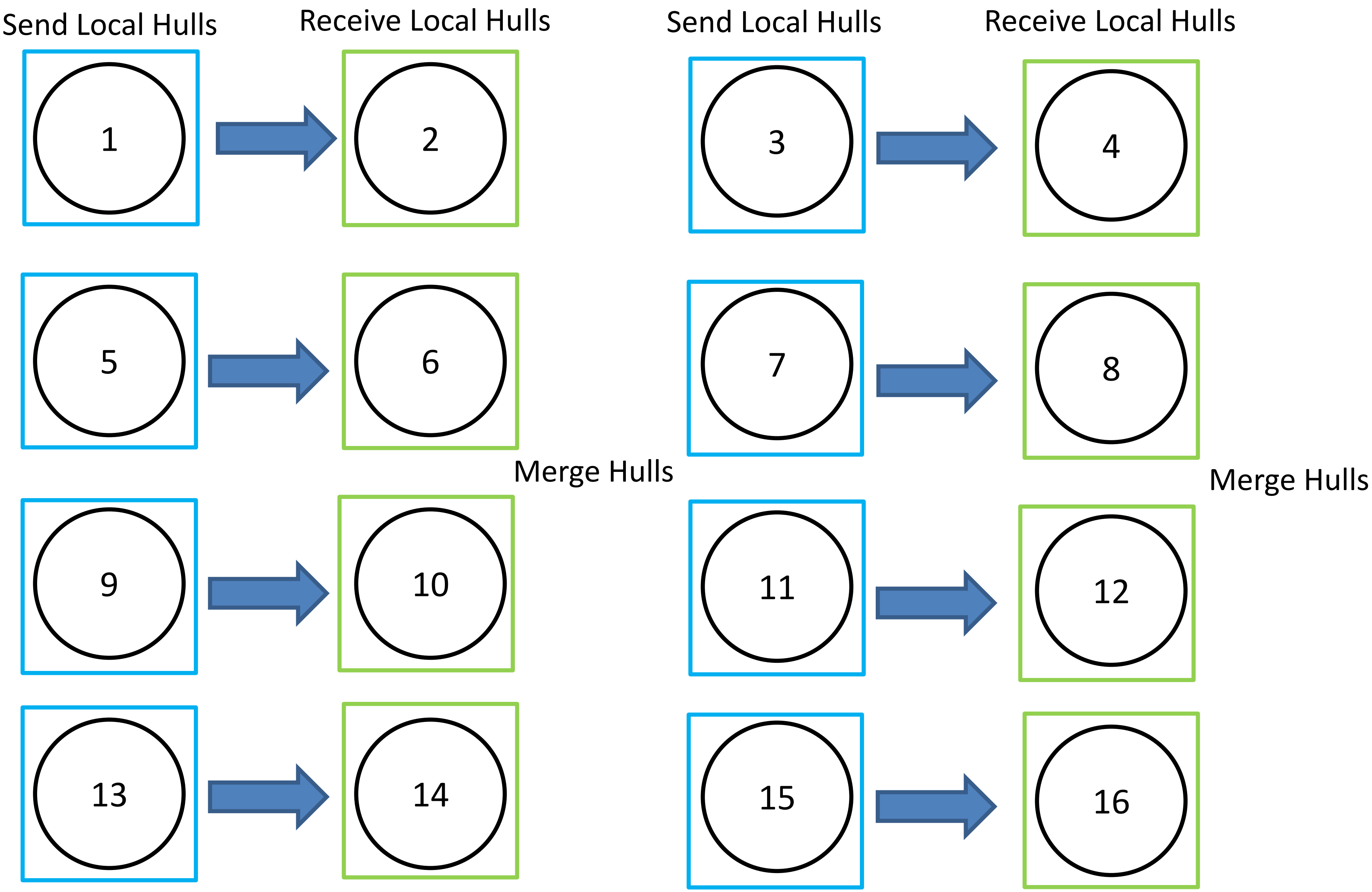
Initial Setup: PEs logically ranked (1 - 16)



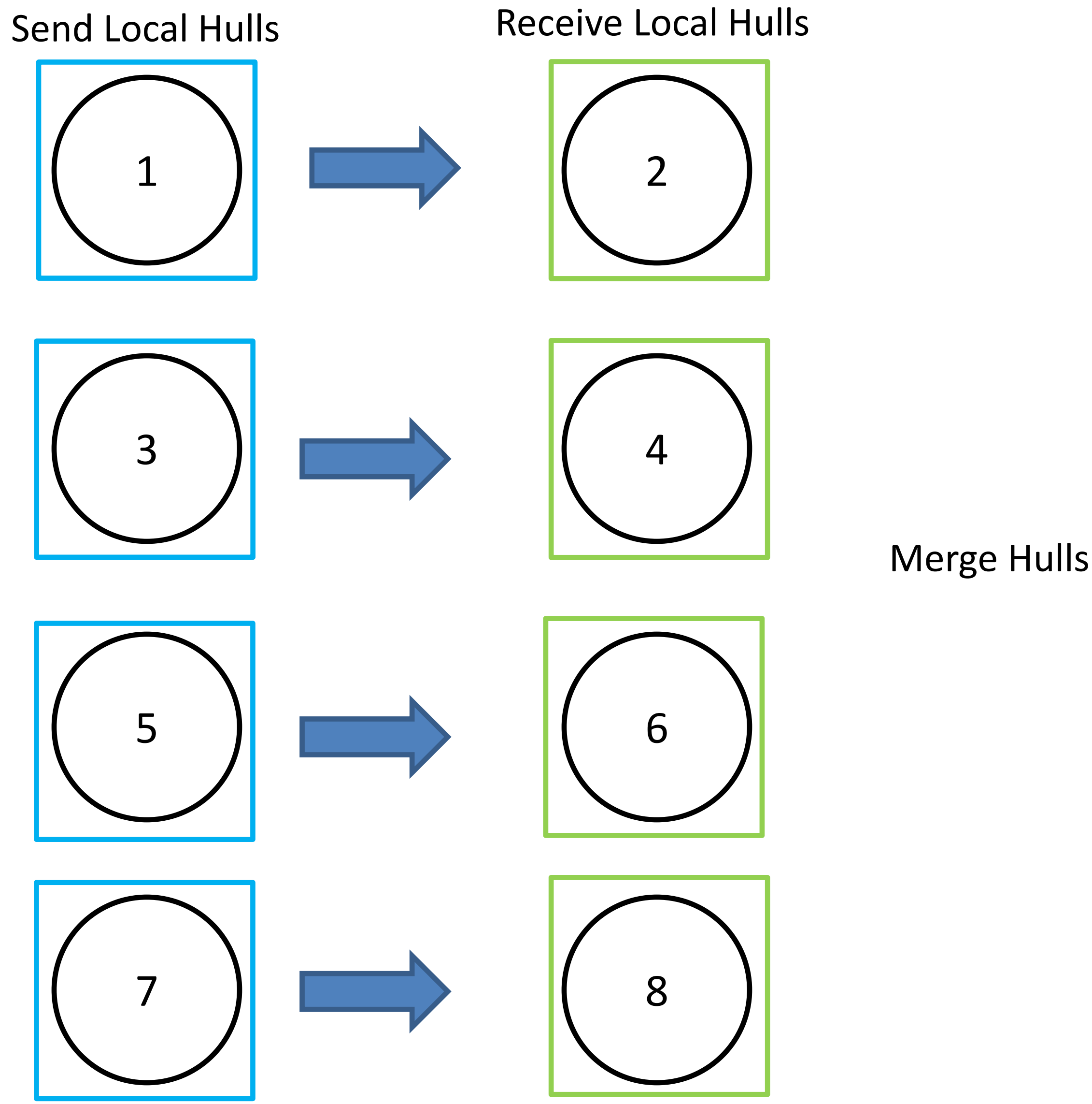
Data Generation and Sequential Computation



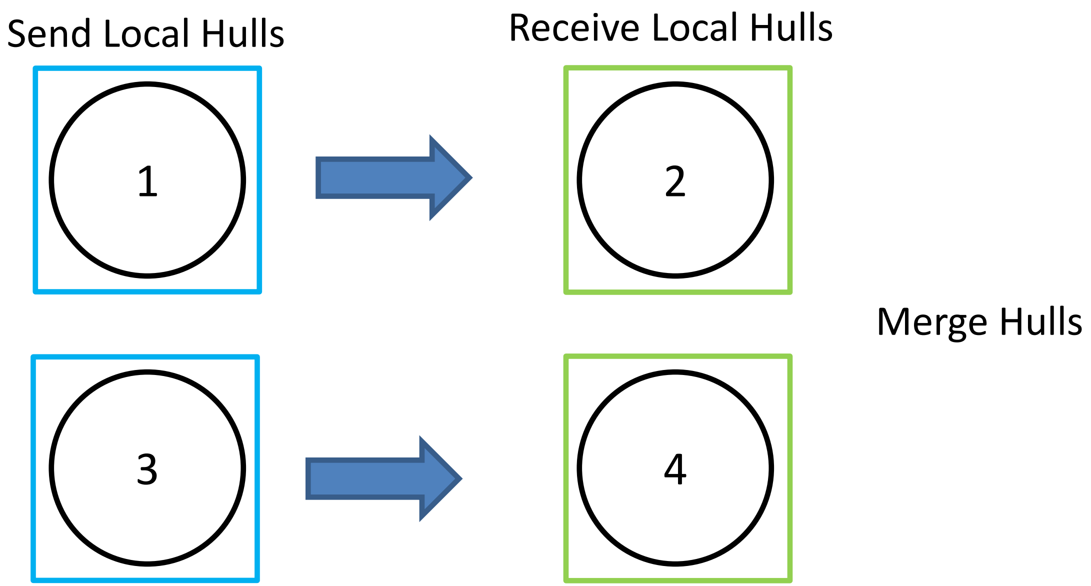
Parallel Merge



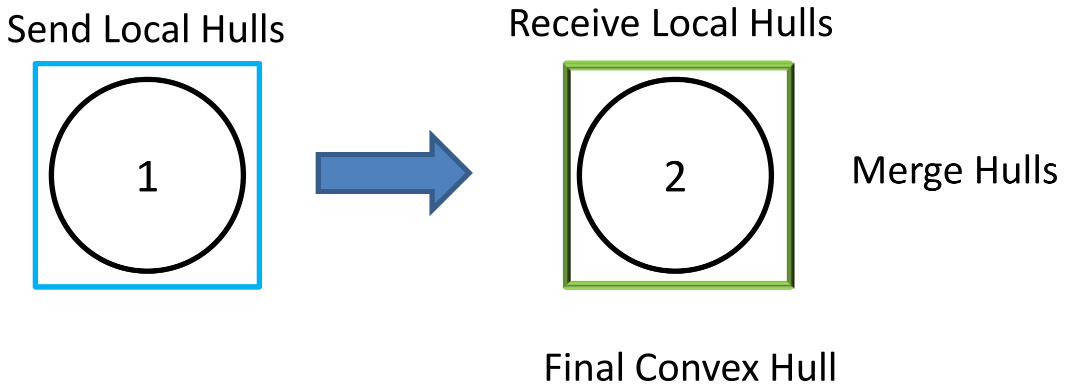
Update Rank (Data in 8 PEs)



Update Rank (Data in 4 PEs)



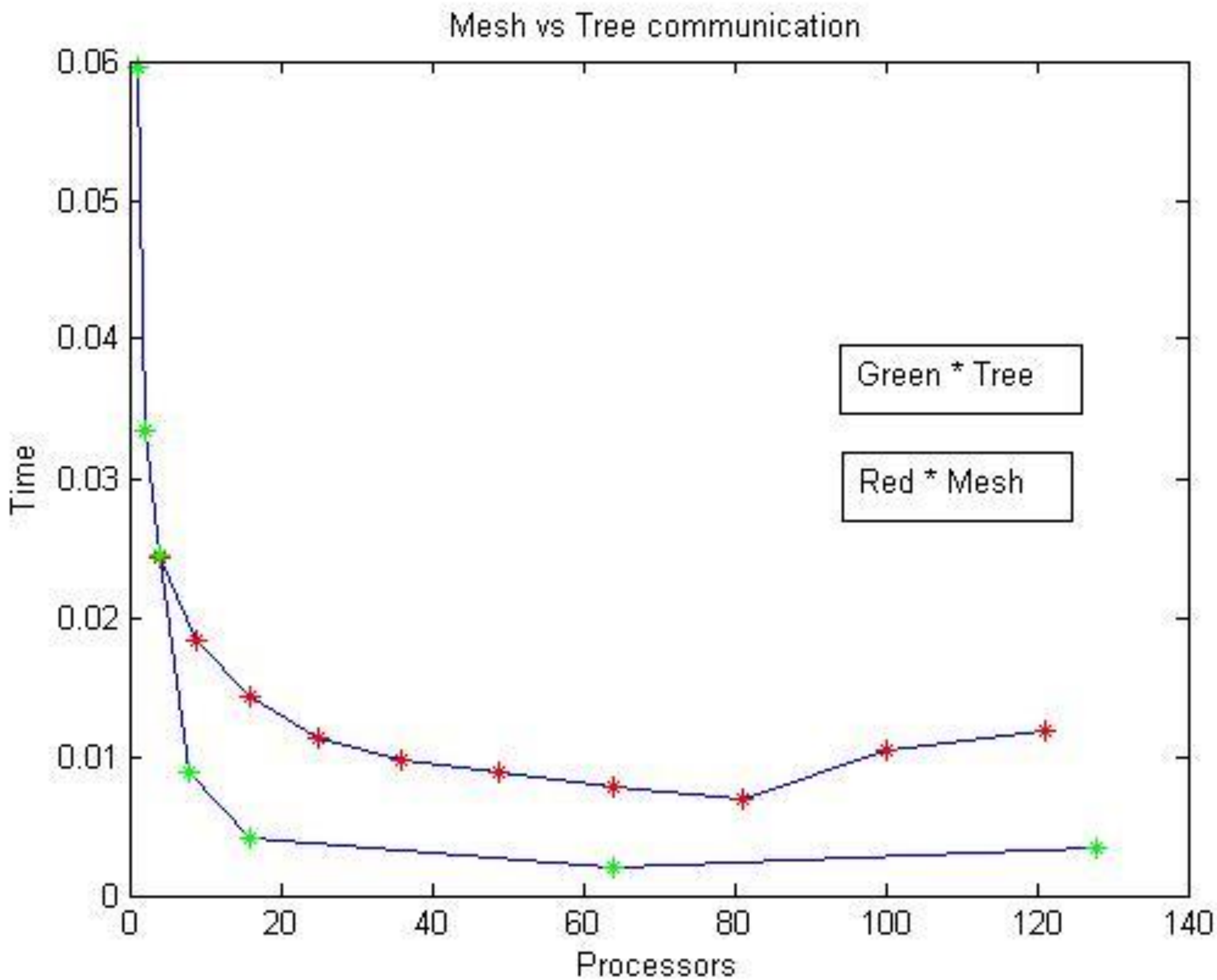
Update Rank (Data in 2 PEs)



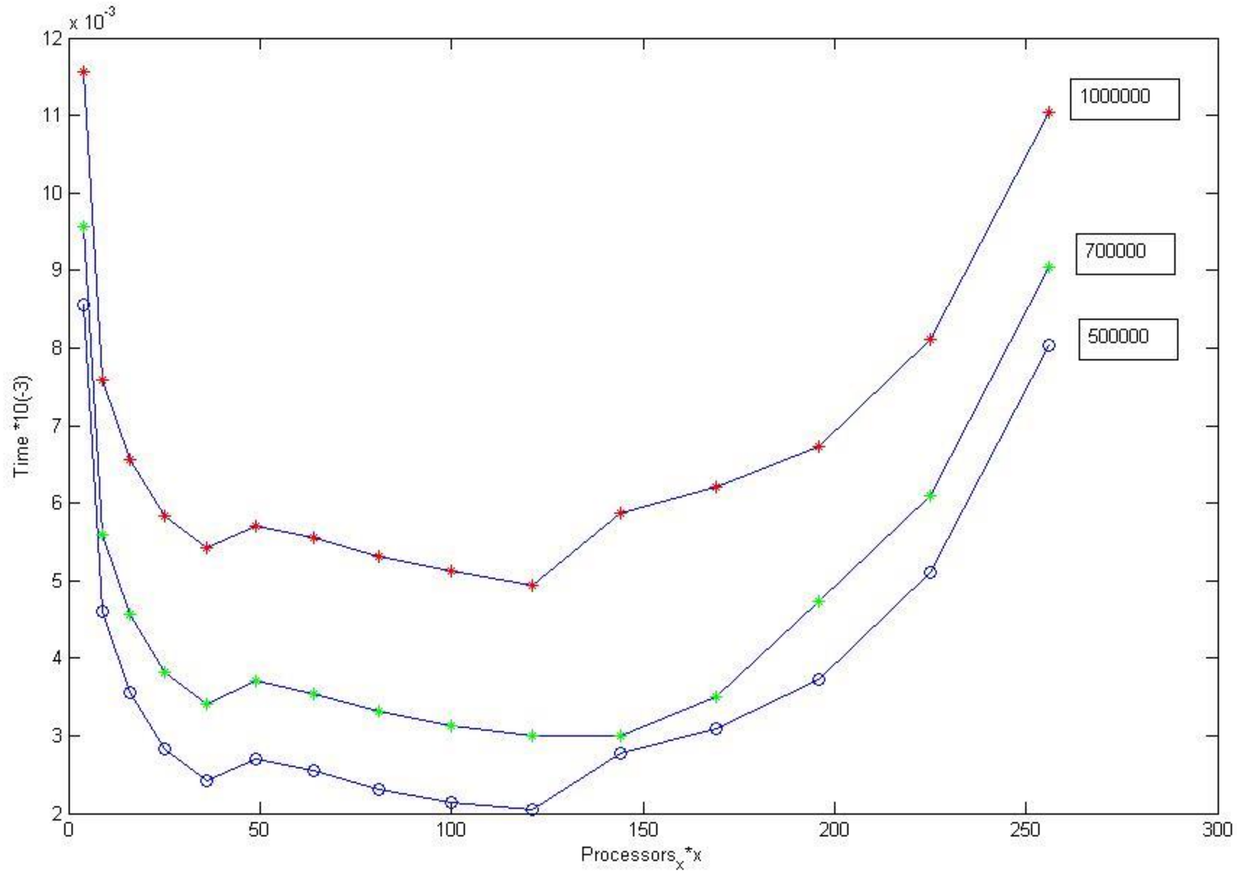
Tree implementation

```
while(stop){
        if(rank%2 == 0)
        {
            count++;
            MPI_Recv(inp, count, MPI_INT, world_rank-iter, 0, MPI_COMM_WORLD,MPI_STATUS_IGNORE);
            get_hull( inp,out,count);

        }
        else if(world_rank!=world_size-1)
        {
            MPI_Send(out, count, MPI_INT, world_rank+iter, 0, MPI_COMM_WORLD);

            stop =0;
        }

                iter = iter*2;

        rank = rank/2;

    }
```

Mesh vs Tree Runtime Analysis



Runtime analysis: Mesh



Reference

- Russ Miller and Laurence boxer; *Algorithms: Sequential and Parallel*; 3rd edition
- Michael T. Goodrich; *Finding the Convex Hull of a Sorted Point Set in Parallel*;
- Russ Miller and Quentin F. Stout; *Efficient Parallel Convex Hull Algorithms*; IEEE Transaction on Computers vol. 37 no. 12

Thank You

Questions???