Data Reduction and Partitioning in an Extreme Scale GPU-Based Clustering Algorithm

Benjamin Welton and Barton Miller

Paradyn Project

University of Wisconsin - Madison

DRBSD-2 Workshop
November 17th 2017
Denver, CO

Our History with Scalable Reductions

Debuggers

Performance Tools

Applications

MRNet

Multicast Reduction Network (MRNet)

[Roth 04]

MRNet was built as a scalable reduction framework for debuggers, performance tools, and applications. Scalability is achieved by a software defined tree based overlay network (TBON).

2004

2017

Our History with Scalable Reductions

Debuggers

Performance Tools

Applications

MRNet

STAT

Stack Trace Analysis Tool
[Arnold 2007]

Live debugging of millions of processes (max run to date 6+ million procs at LLNL). MRNet provides the scaleable reduction of stack traces collected from processes.

2004

2017

Our History with Scalable Reductions

Debuggers

Performance Tools

Applications

MRNet

STAT

Cray CCBD

Total View

Cray ATP

Since STAT's release, MRNet has been used as the reduction framework by a number of commercial debuggers.

2004

2017

Our History with Scalable Reductions

Debuggers

STAT

Cray CCBD

Total View

Cray ATP

Performance Tools

MRNet

TAU

TAU
[Nataraj 2008]

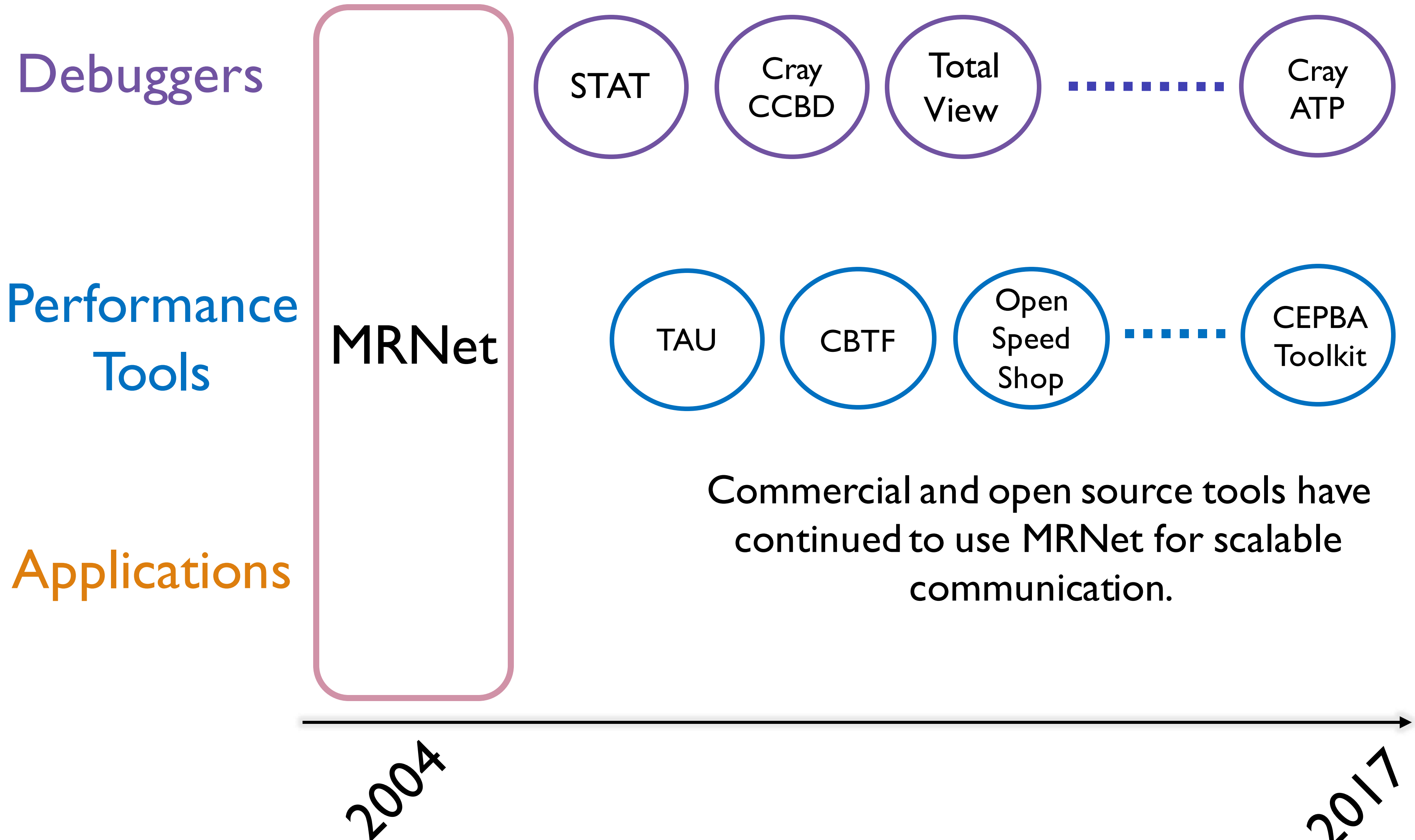
Performance monitoring framework. MRNet is used to scalable collect performance data from thousands of nodes.

Applications

2004

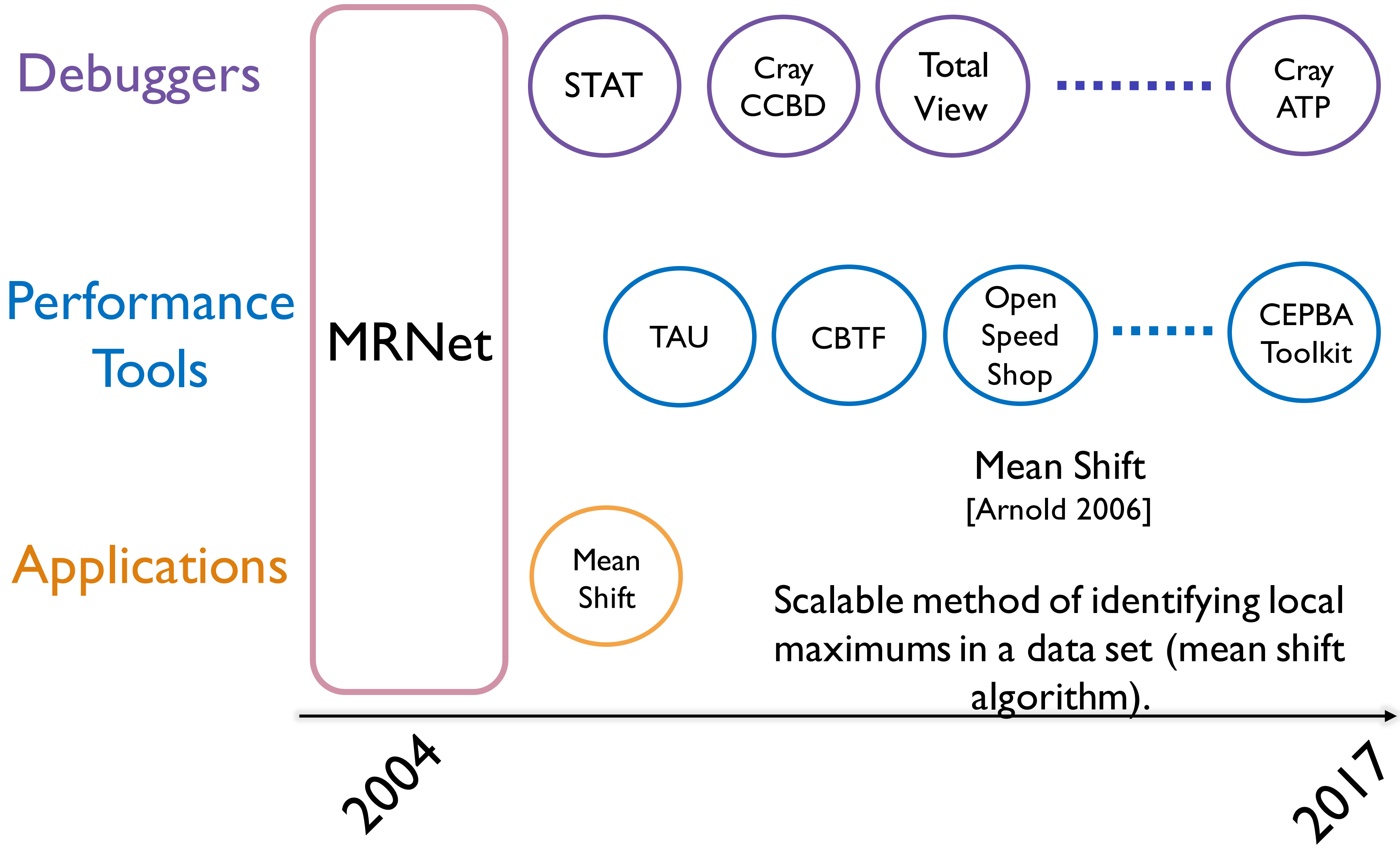
2017

Our History with Scalable Reductions



Commercial and open source tools have continued to use MRNet for scalable communication.

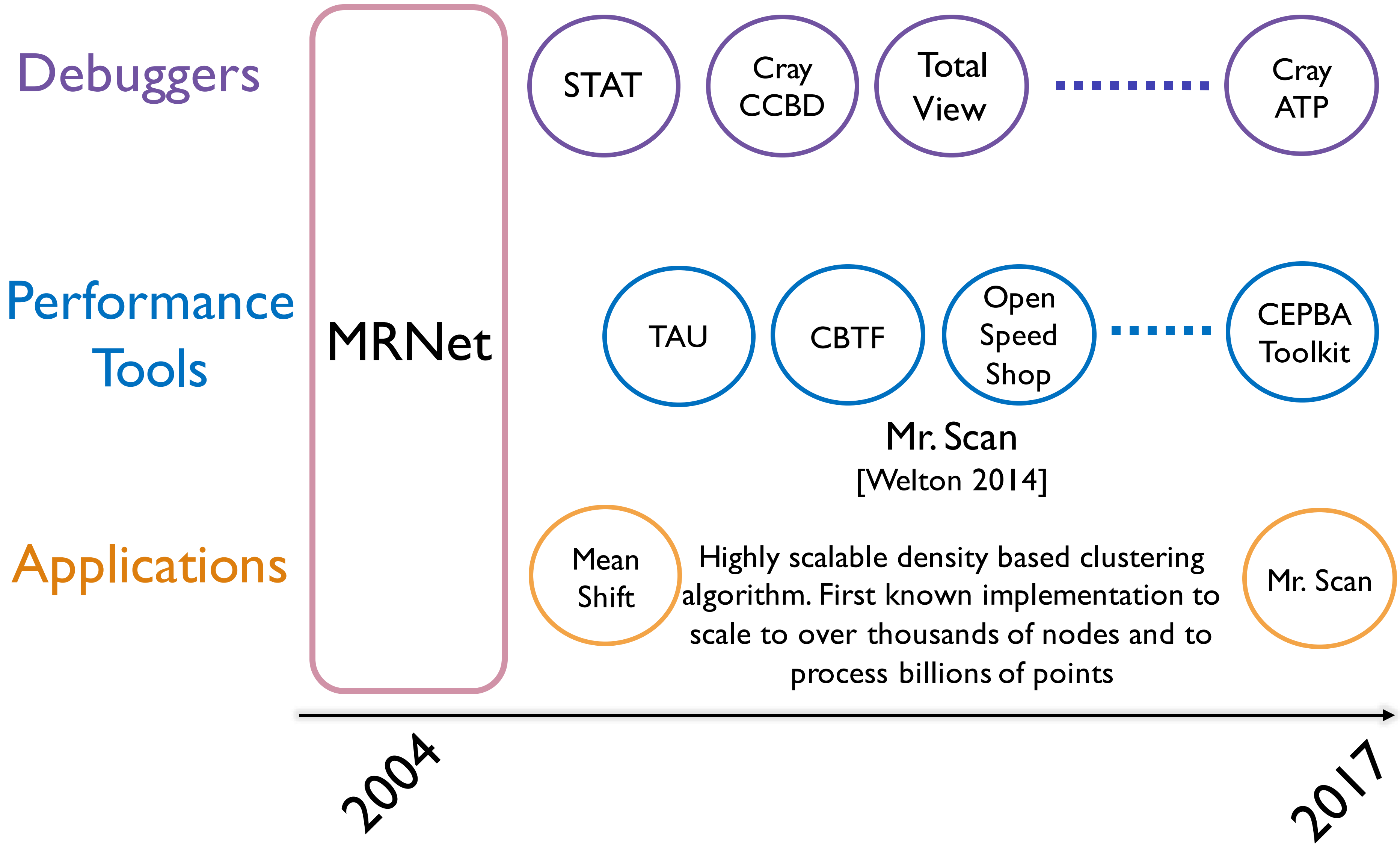
Our History with Scalable Reductions



Mean Shift
[Arnold 2006]

Scalable method of identifying local maximums in a data set (mean shift algorithm).

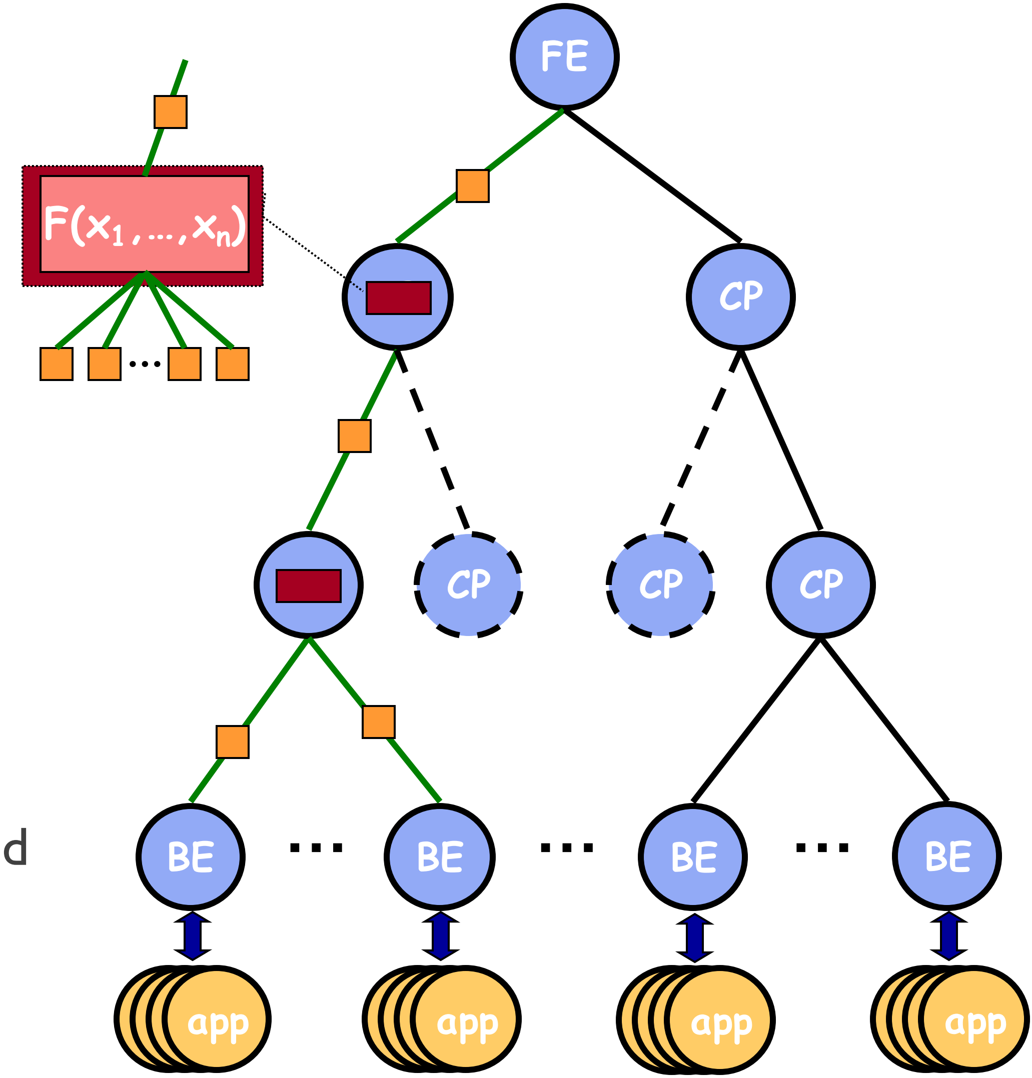
Our History with Scalable Reductions



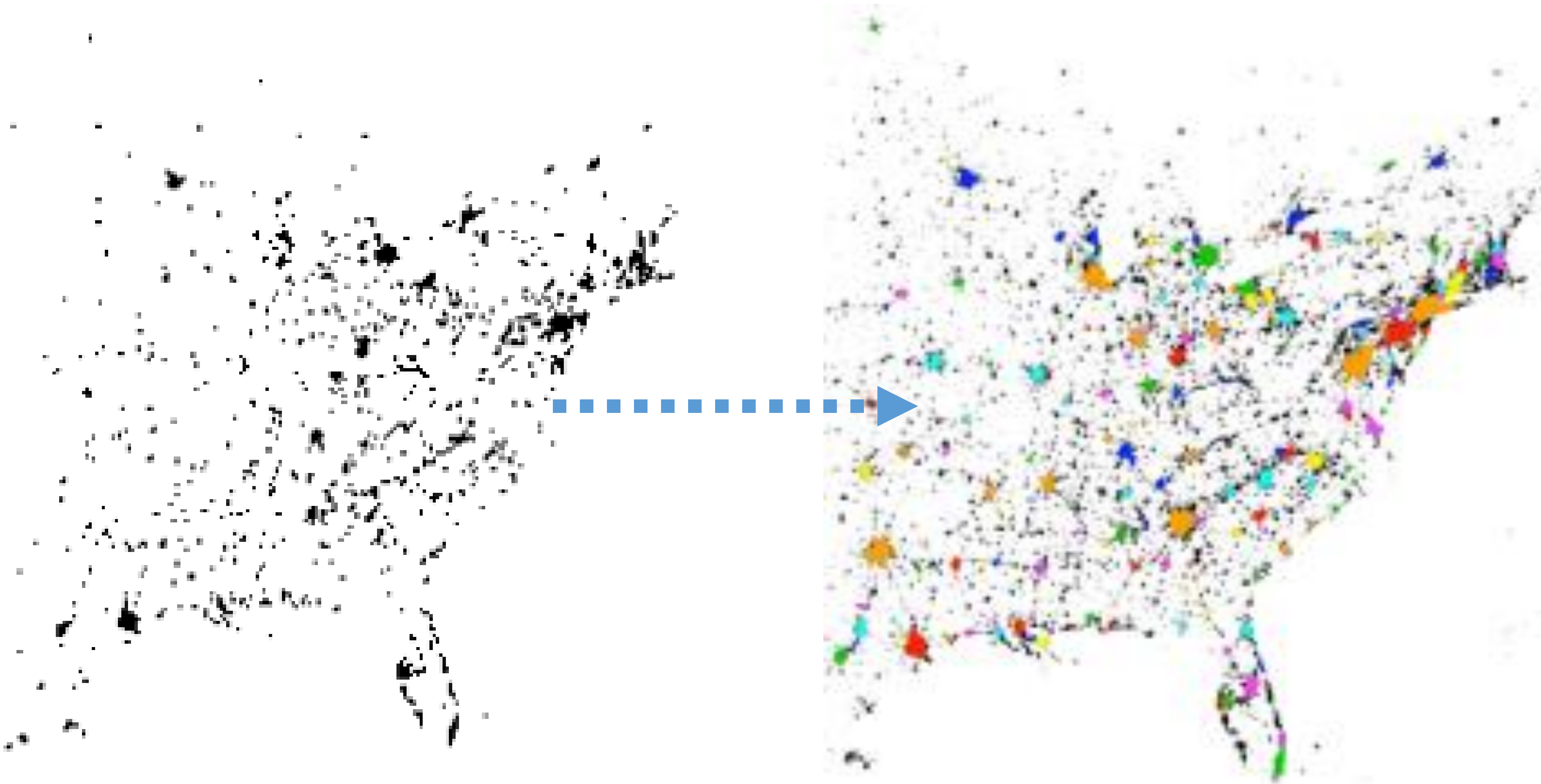
MRNet – Multicast / Reduction Network

General-purpose Tree Based Overlay Network (TBON)

- Network: user-defined topology
- Stream: logical data channel
  - to a set of back-ends
  - multicast, gather, and custom reduction
- Packet: collection of data
- Filter: stream data operator
  - User definable aggregation and reduction operations.
- Fully asynchronous across multiple streams.



Introducing Mr. Scan

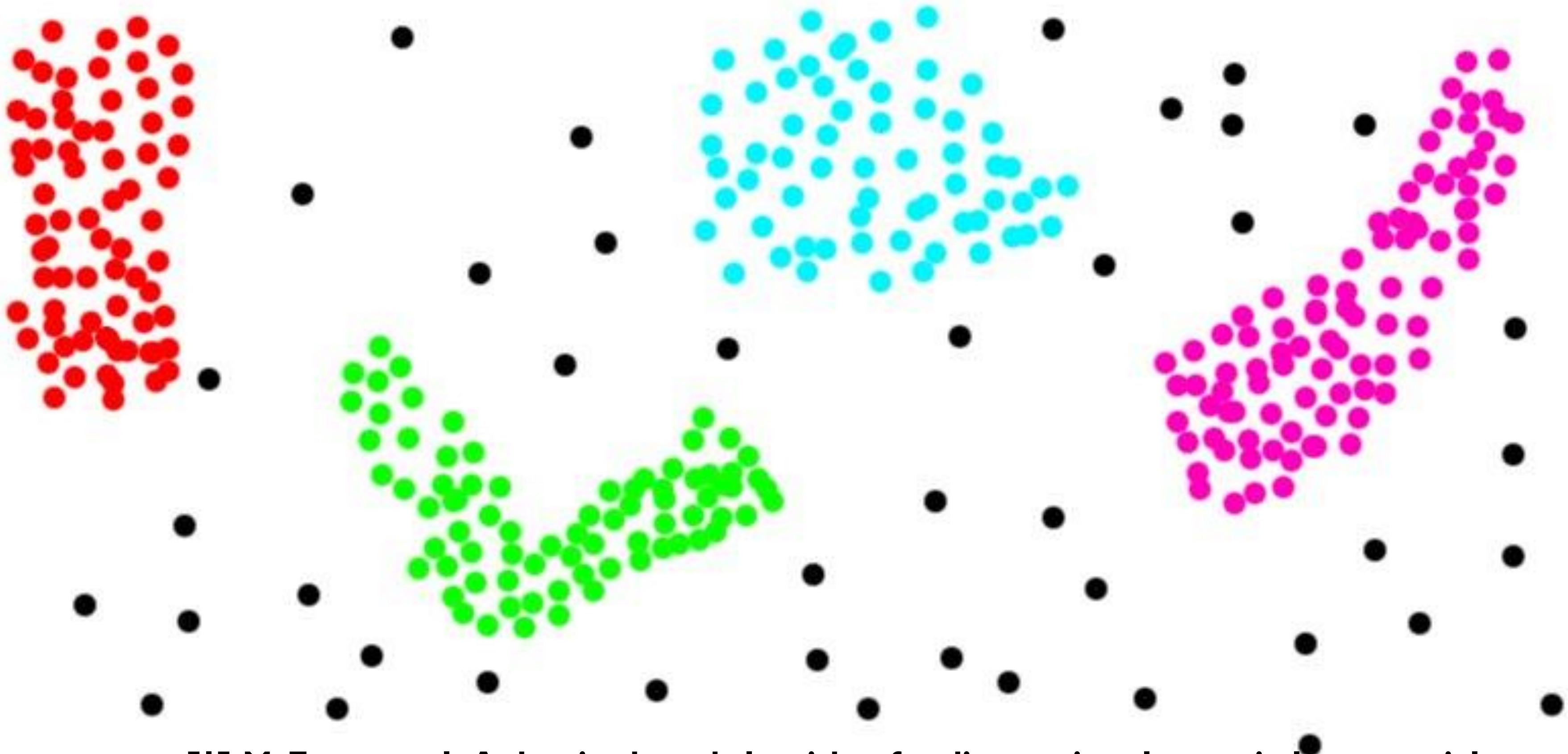


Mr. Scan is a scalable density-based clustering algorithm

Designed to cluster billions of data points.

Clustering Example (DBSCAN[1])

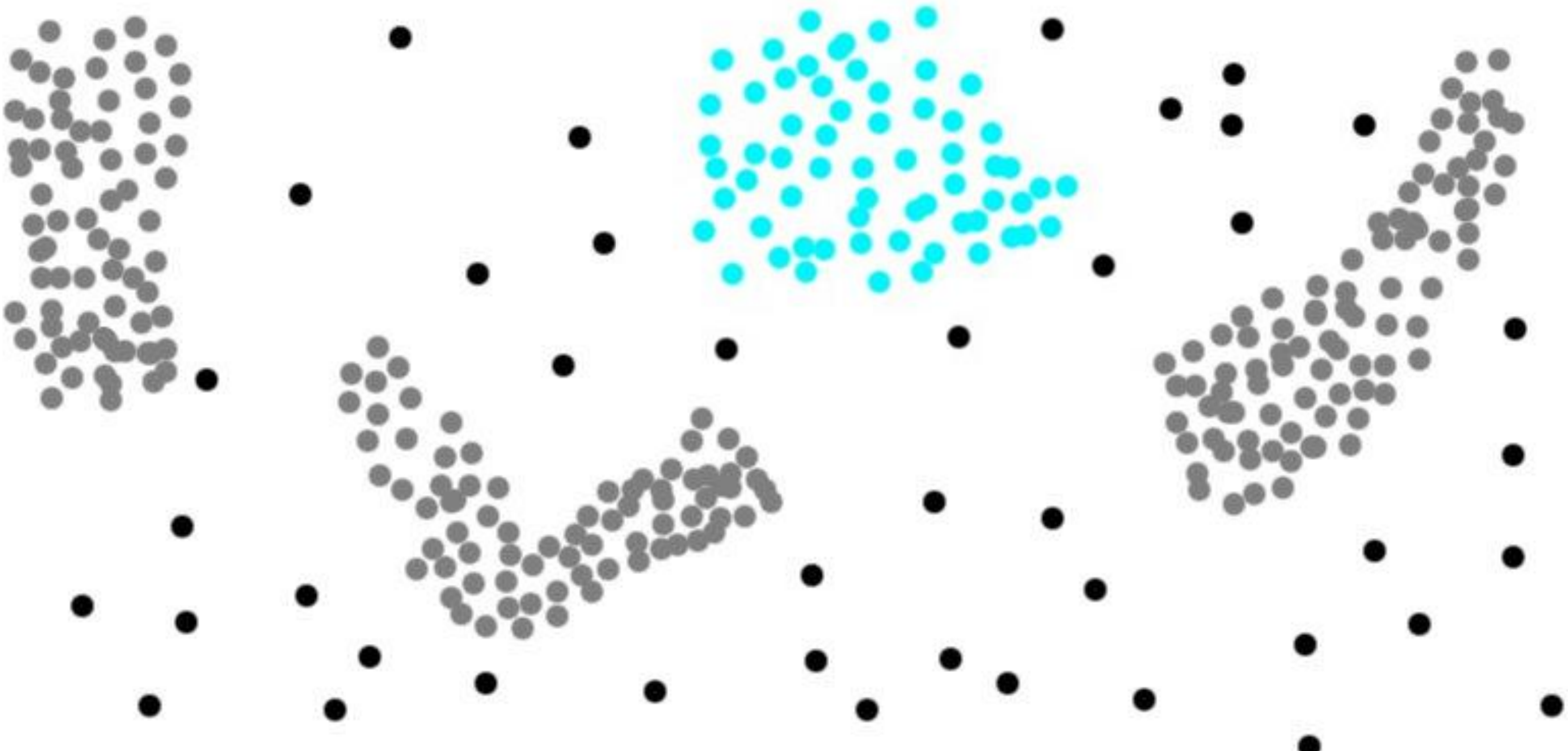
Goal: Find regions that meet minimum density and spatial distance characteristics



[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Clustering Example (DBSCAN[1])

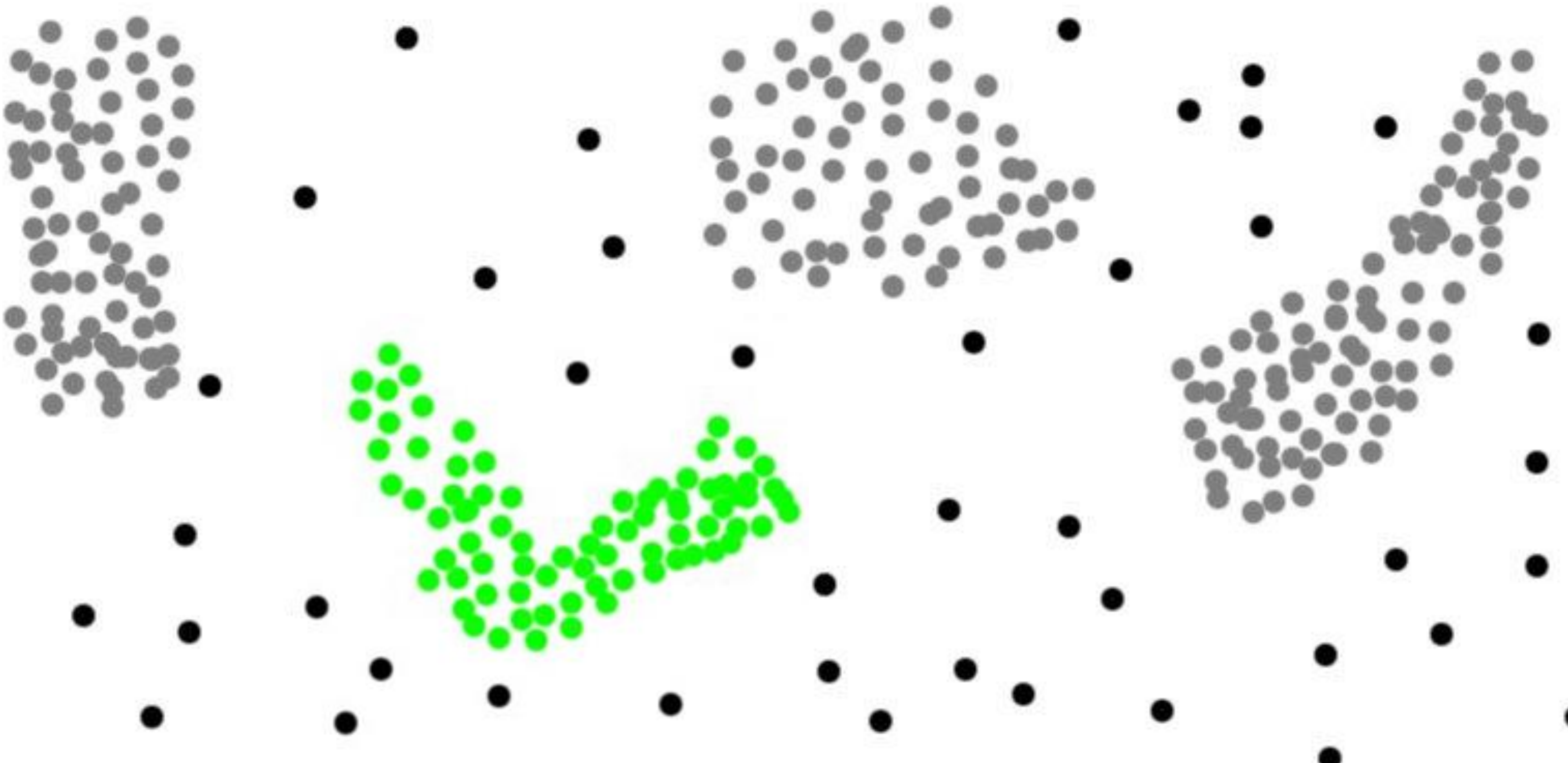
Goal: Find regions that meet minimum density and spatial distance characteristics



[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Clustering Example (DBSCAN[1])

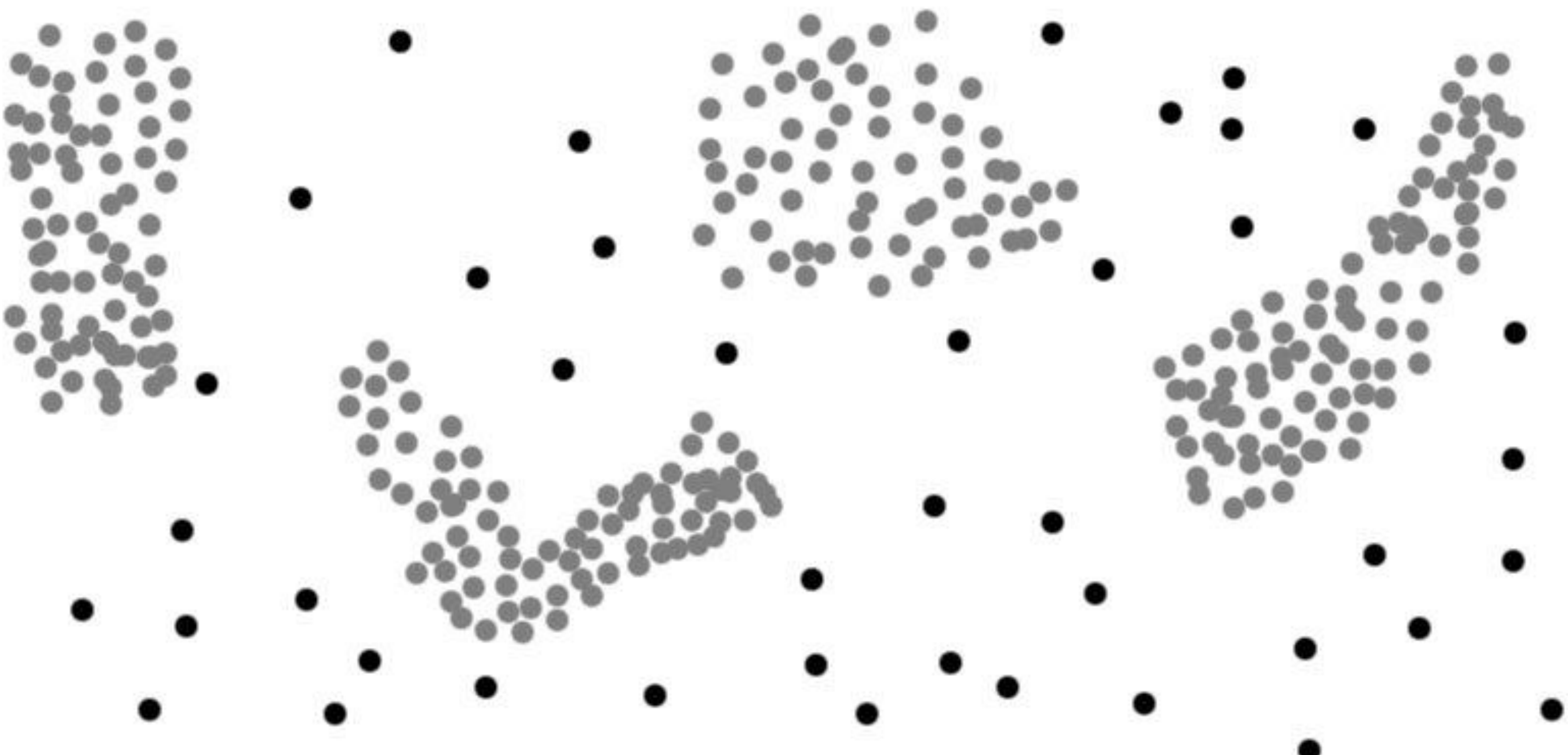
Goal: Find regions that meet minimum density and spatial distance characteristics



[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Clustering Example (DBSCAN[1])

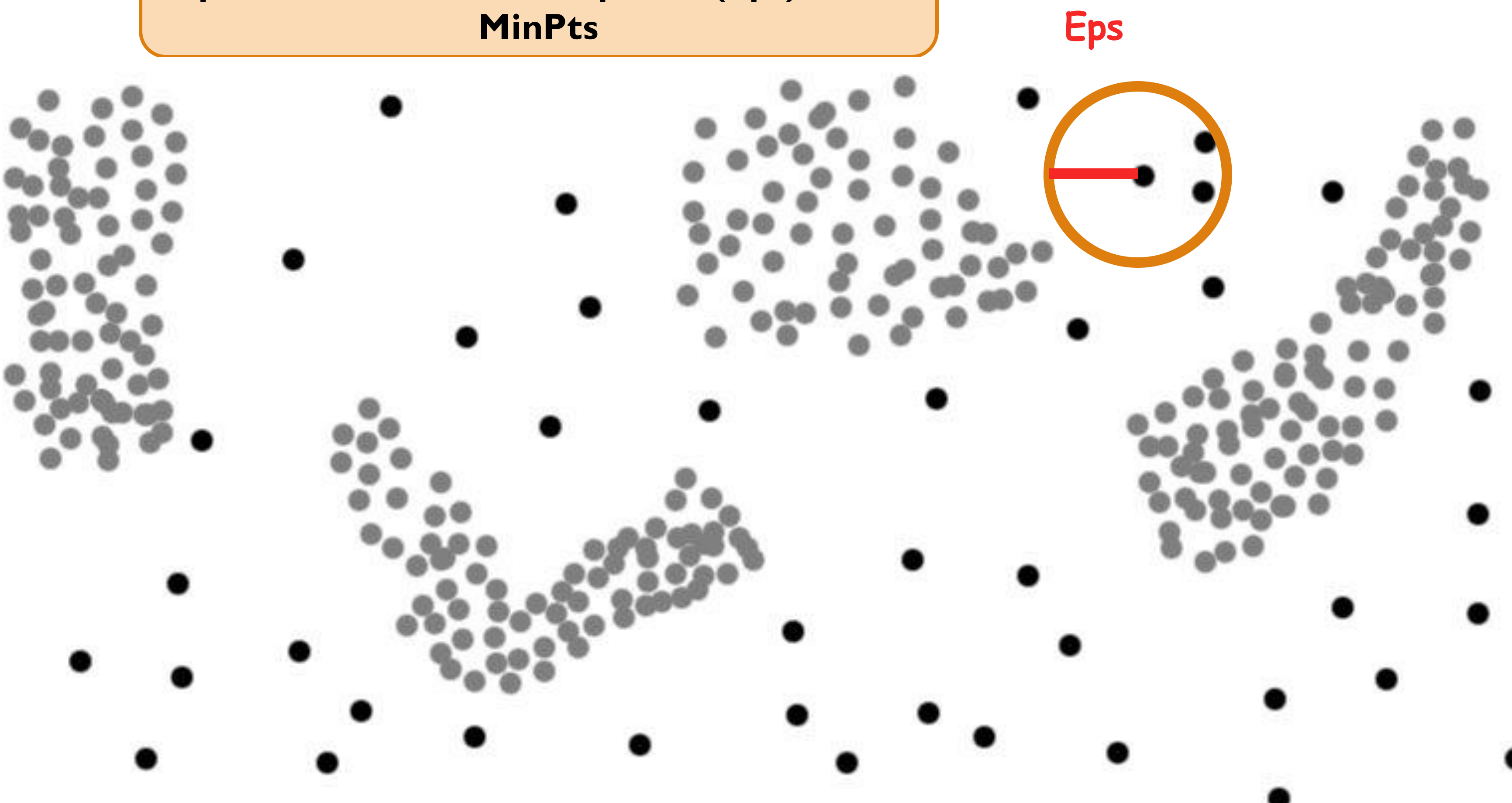
The two parameters that determine if a point is in a cluster is Epsilon (Eps), and MinPts



[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Clustering Example (DBSCAN[1])

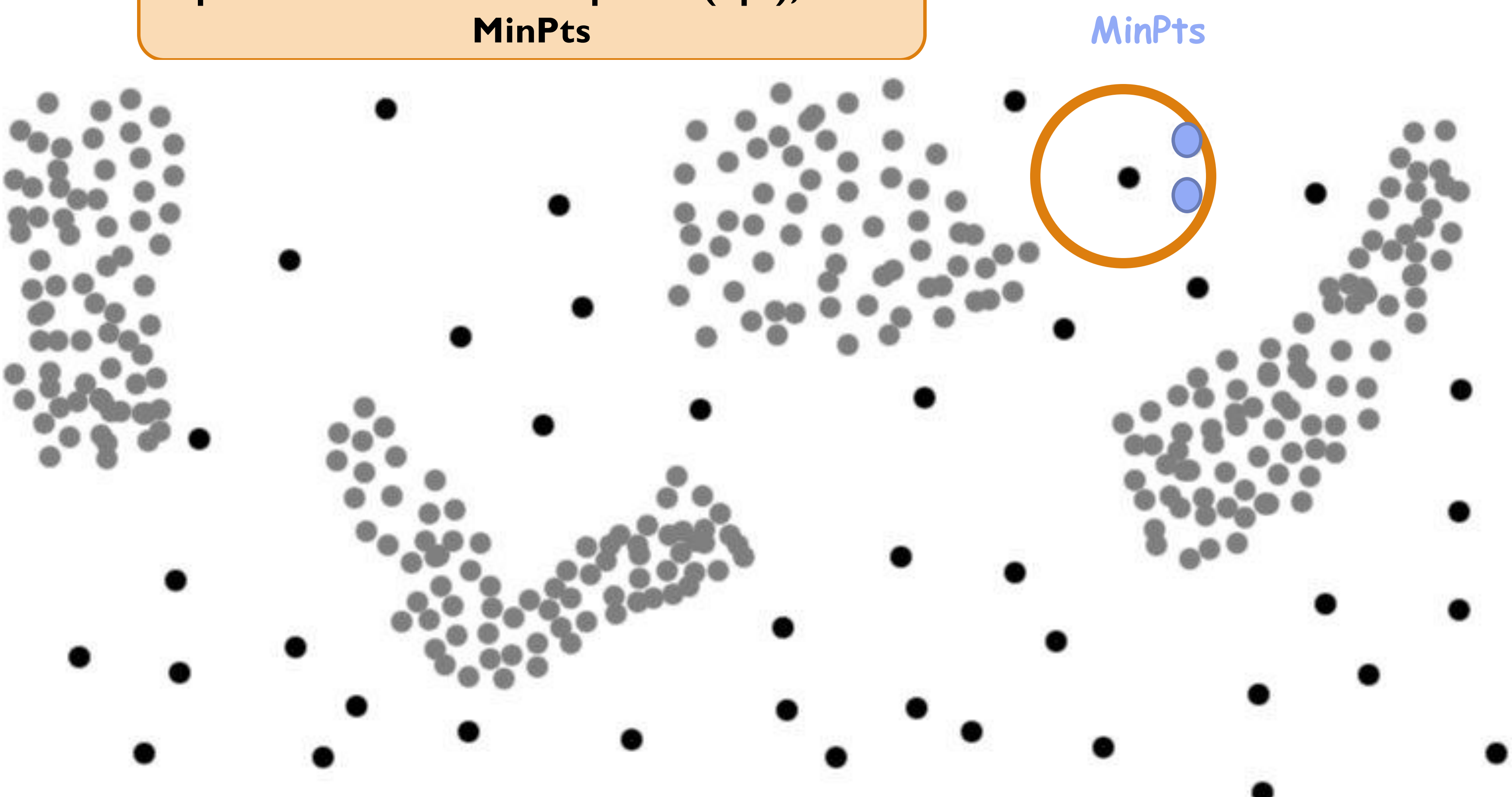
The two parameters that determine if a point is in a cluster is Epsilon (Eps), and MinPts



[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Clustering Example (DBSCAN[1])

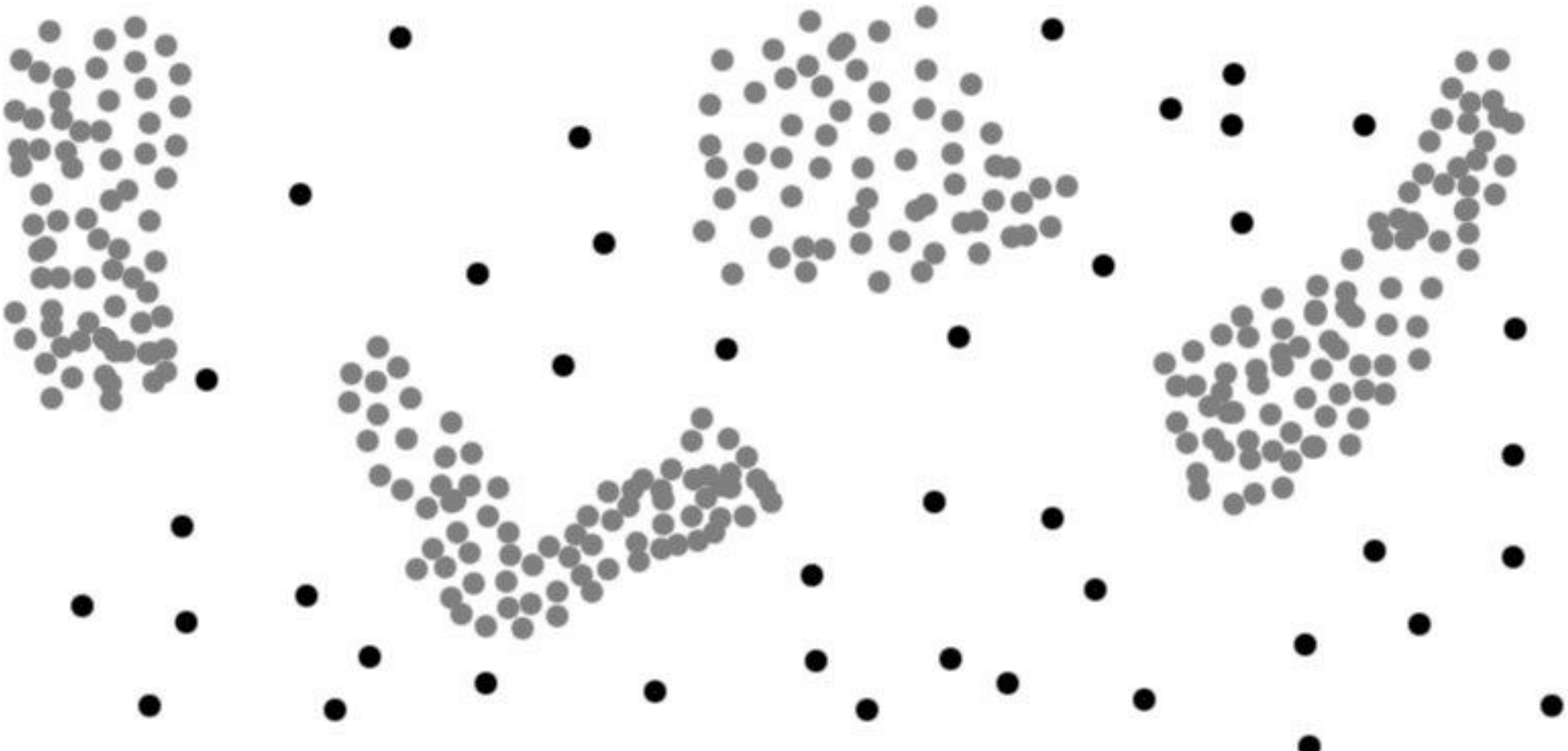
The two parameters that determine if a point is in a cluster is Epsilon (Eps), and MinPts



[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Clustering Example (DBSCAN[1])

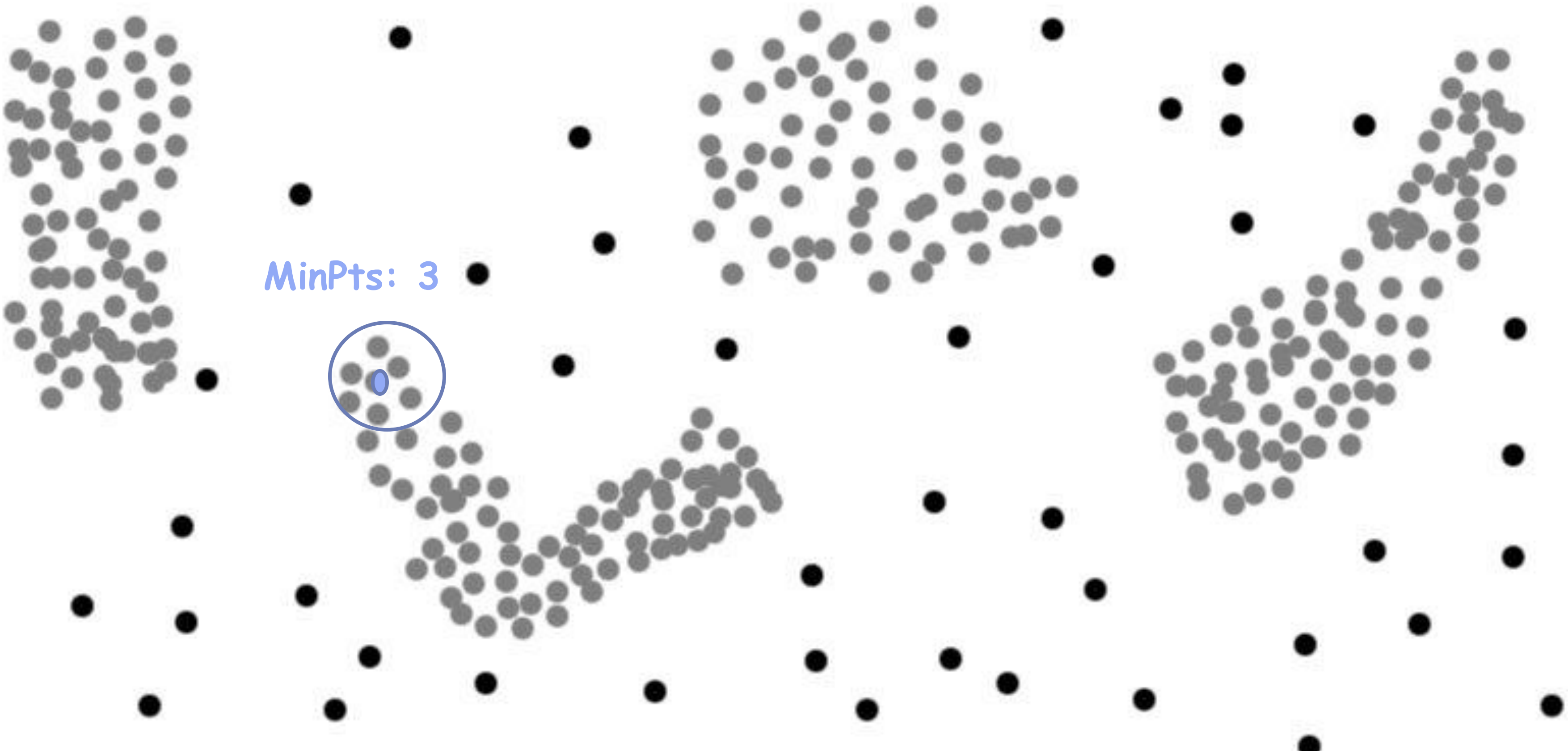
If the number of points in Eps is > MinPts, the point is a core point.



[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Clustering Example (DBSCAN[1])

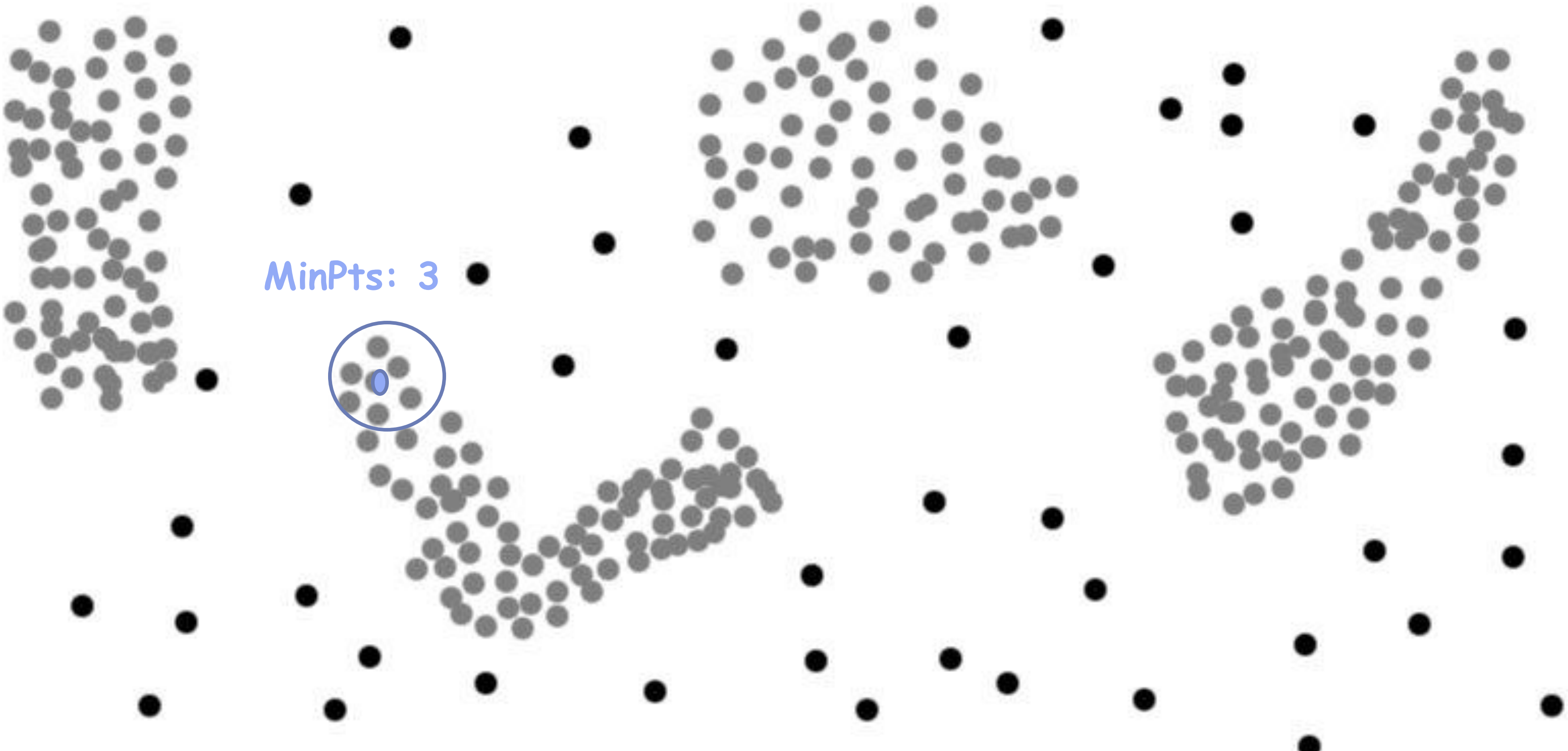
If the number of points in Eps is > MinPts, the point is a core point.



[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Clustering Example (DBSCAN[1])

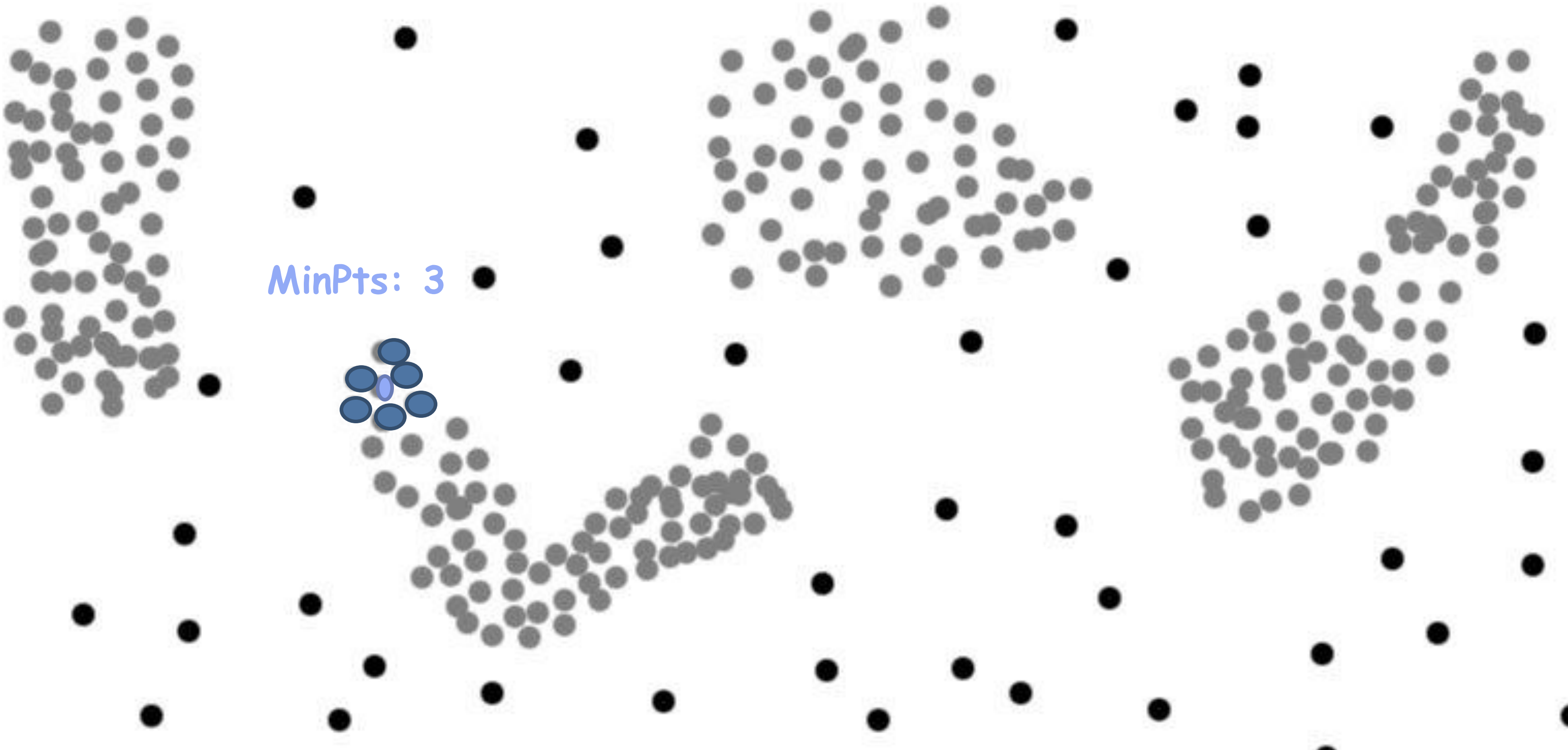
For every discovered point, this same calculation is performed until the cluster is fully expanded



[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Clustering Example (DBSCAN[1])

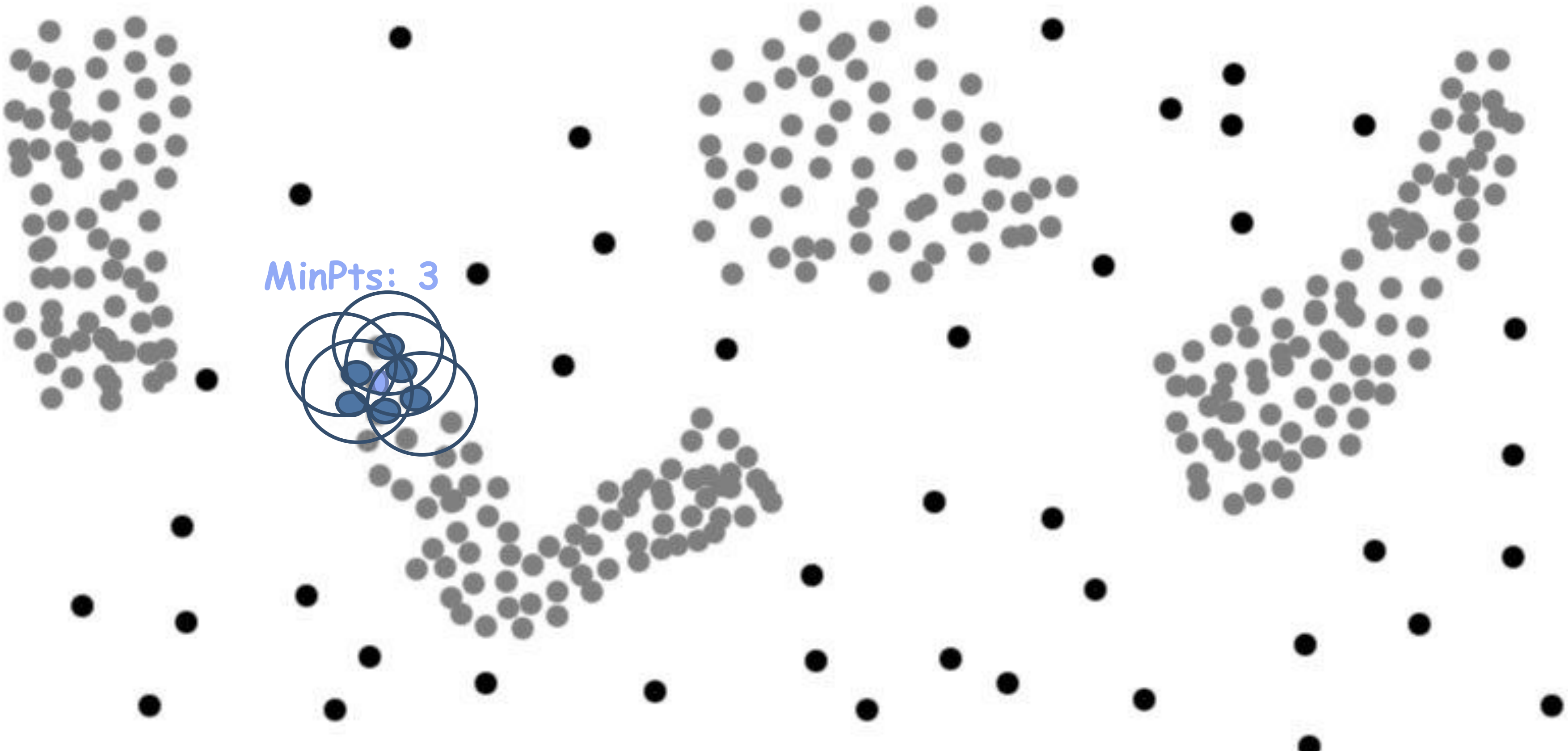
For every discovered point, this same calculation is performed until the cluster is fully expanded



[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Clustering Example (DBSCAN[1])

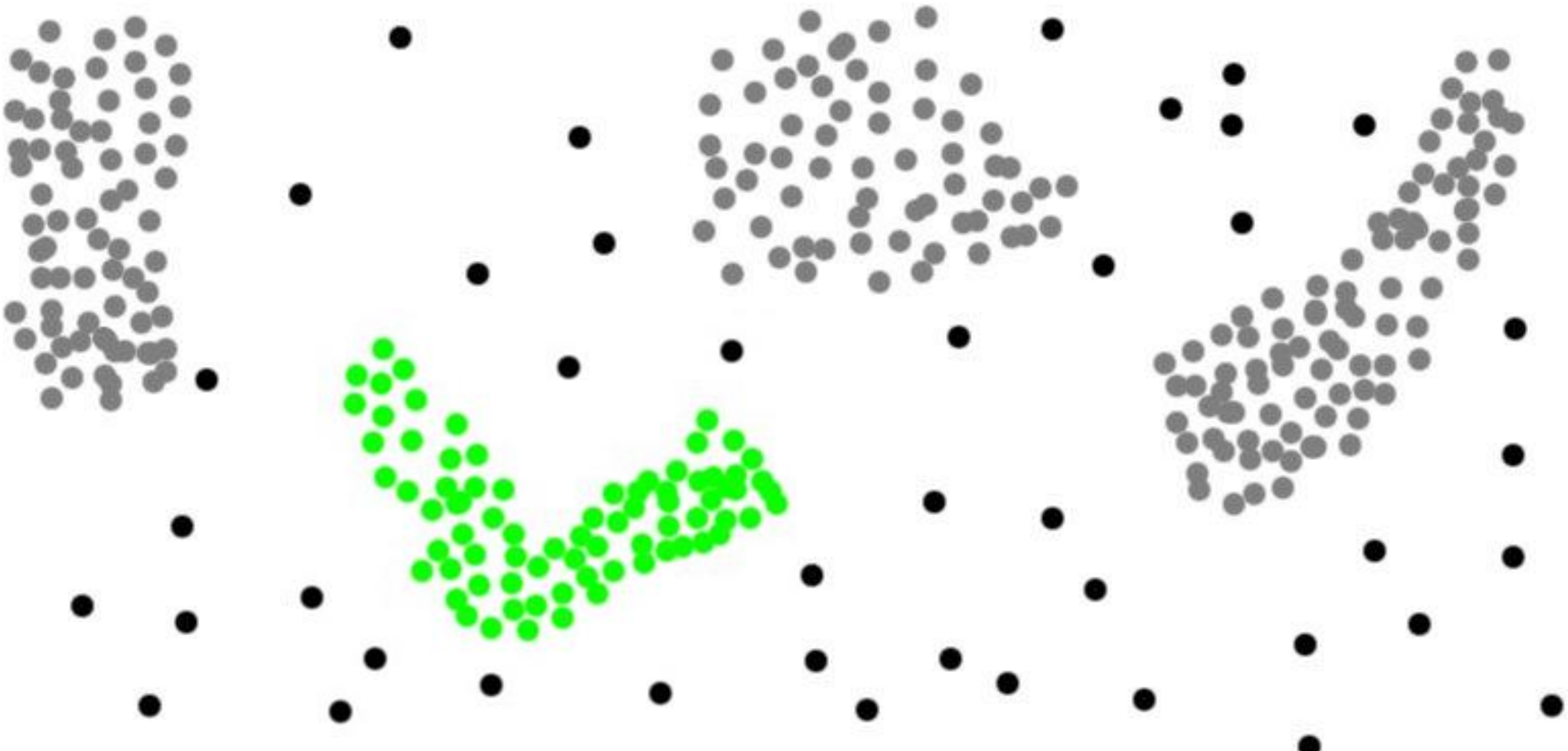
For every discovered point, this same calculation is performed until the cluster is fully expanded



[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Clustering Example (DBSCAN[1])

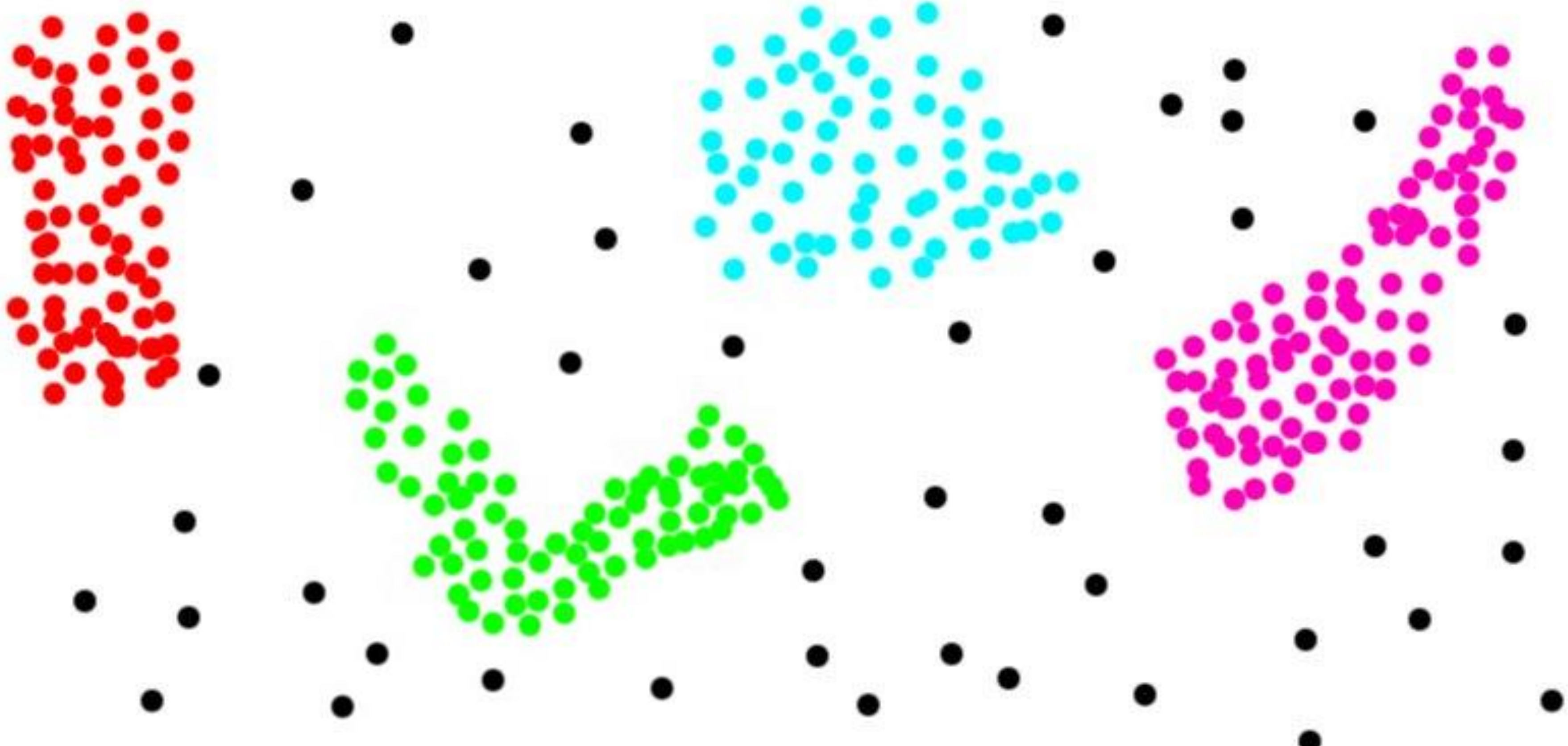
For every discovered point, this same calculation is performed until the cluster is fully expanded



[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Clustering Example (DBSCAN[1])

For every discovered point, this same calculation is performed until the cluster is fully expanded



[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Intro to Mr. Scan

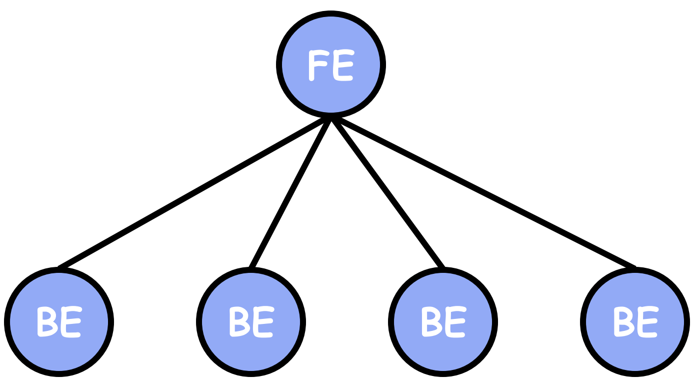
<u>Mr. Scan Phases</u>

→ **Partition: Distributed**

DBSCAN: GPU (on BE)

Merge: CPU (x #levels)

Sweep: CPU (x #levels)



Intro to Mr. Scan

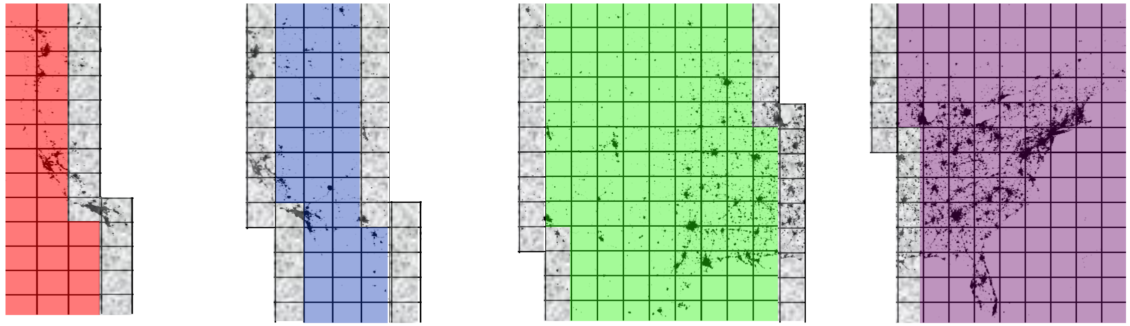
Mr. Scan Phases

→ **Partition: Distributed**

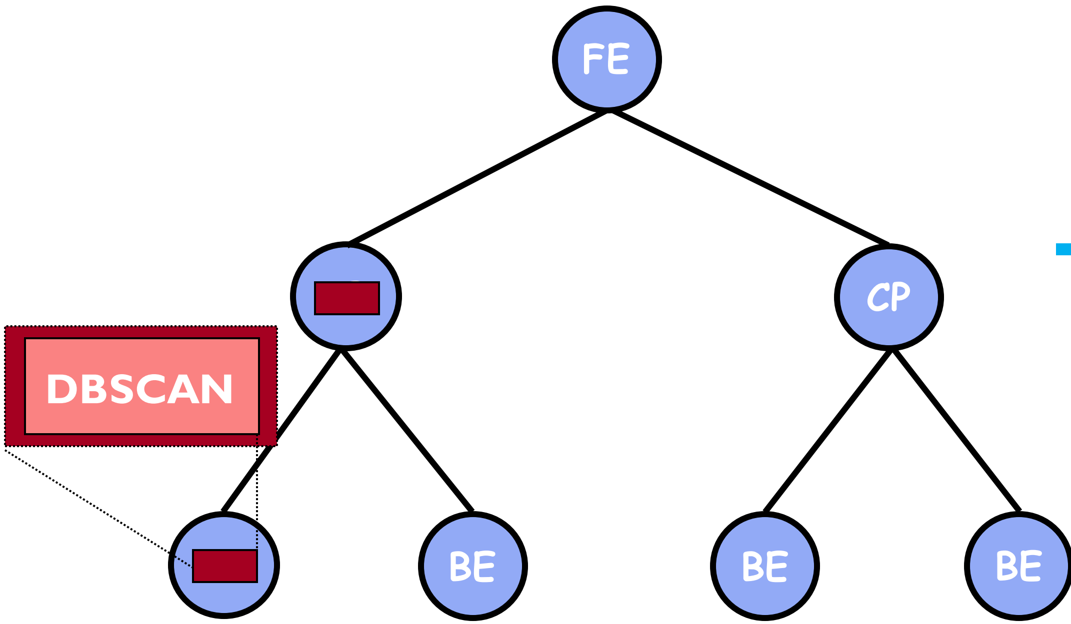
DBSCAN: GPU (on BE)

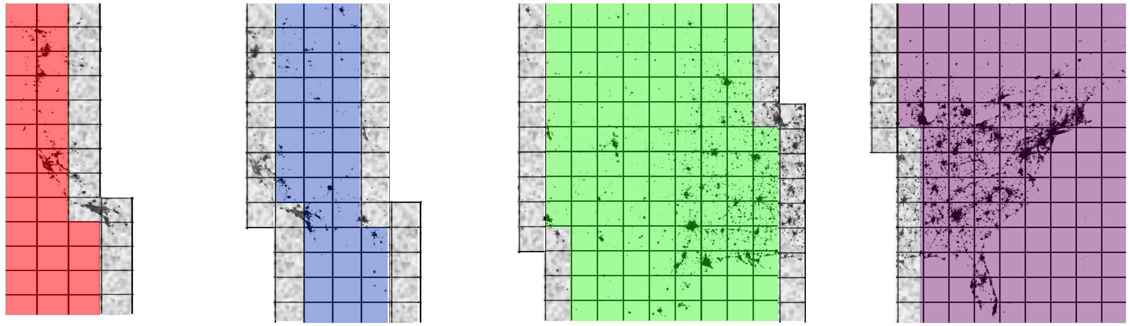
Merge: CPU (x #levels)

Sweep: CPU (x #levels)



Intro to Mr. Scan





Mr. Scan Phases

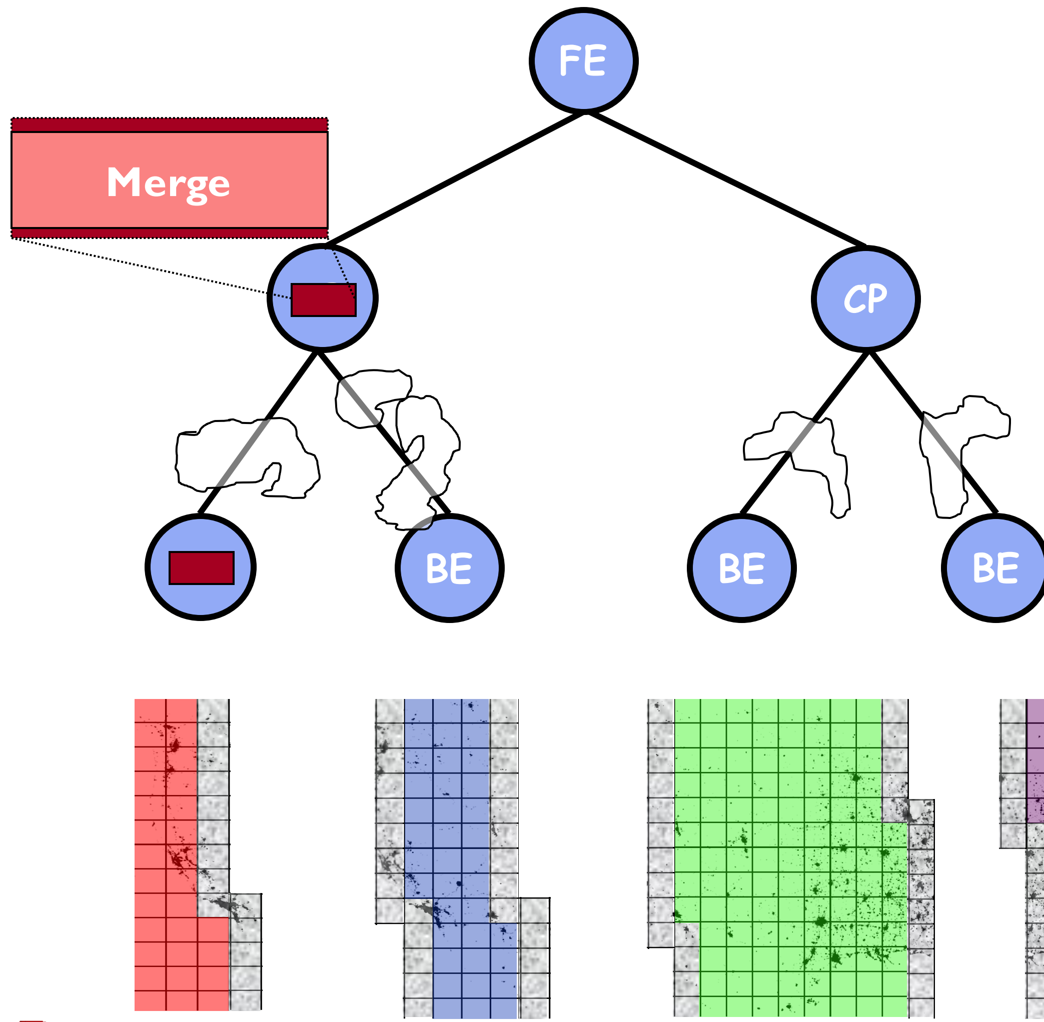
Partition: Distributed

DBSCAN: GPU (on BE)

Merge: CPU (x #levels)

Sweep: CPU (x #levels)

Intro to Mr. Scan



<u>Mr. Scan Phases</u>

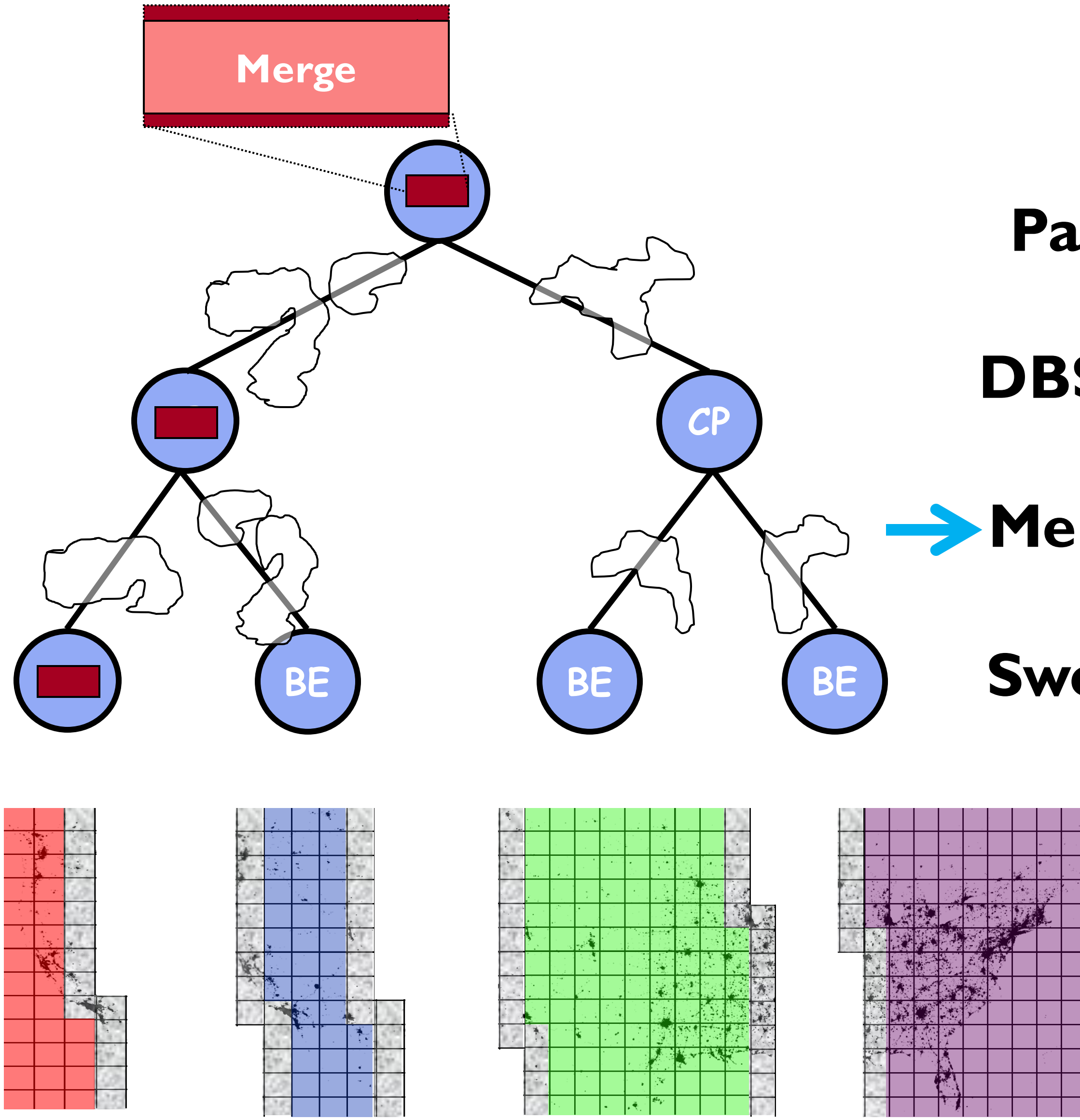
Partition: Distributed

DBSCAN: GPU (on BE)

Merge: CPU (x #levels)

Sweep: CPU (x #levels)

Intro to Mr. Scan



Mr. Scan Phases

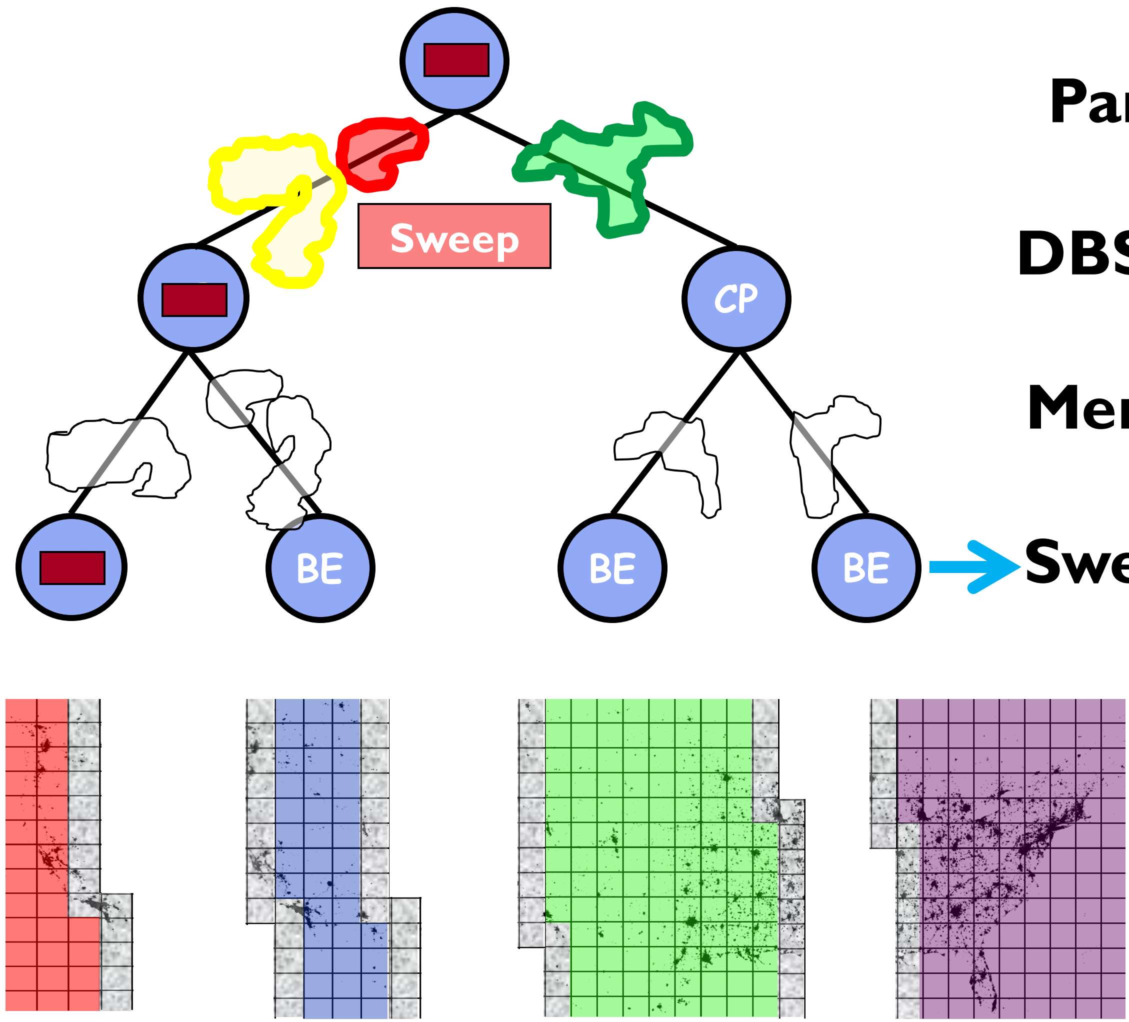
Partition: Distributed

DBSCAN: GPU (on BE)

Merge: CPU (x #levels)

Sweep: CPU (x #levels)

Intro to Mr. Scan



Mr. Scan Phases

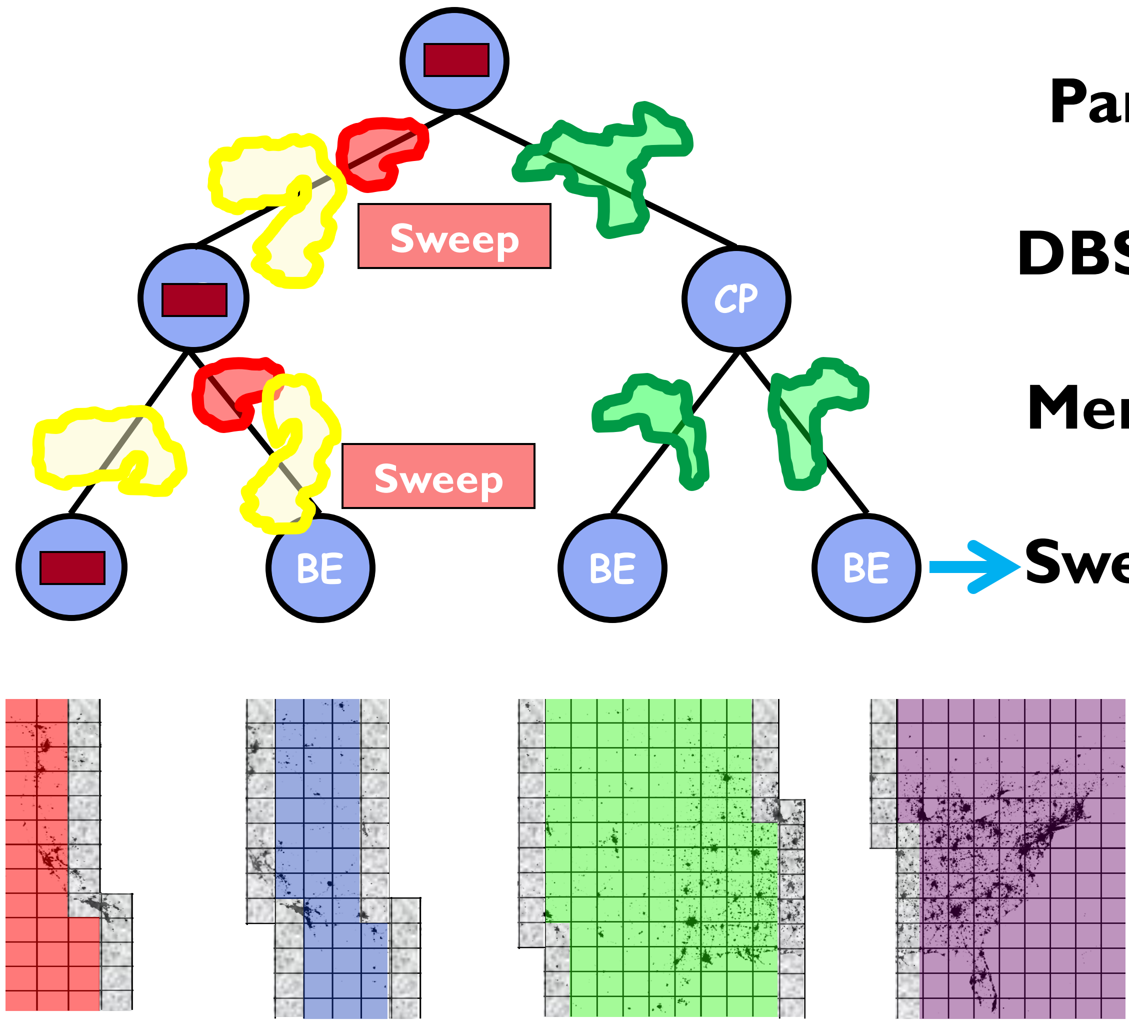
Partition: Distributed

DBSCAN: GPU (on BE)

Merge: CPU (x #levels)

Sweep: CPU (x #levels)

Intro to Mr. Scan



Mr. Scan Phases

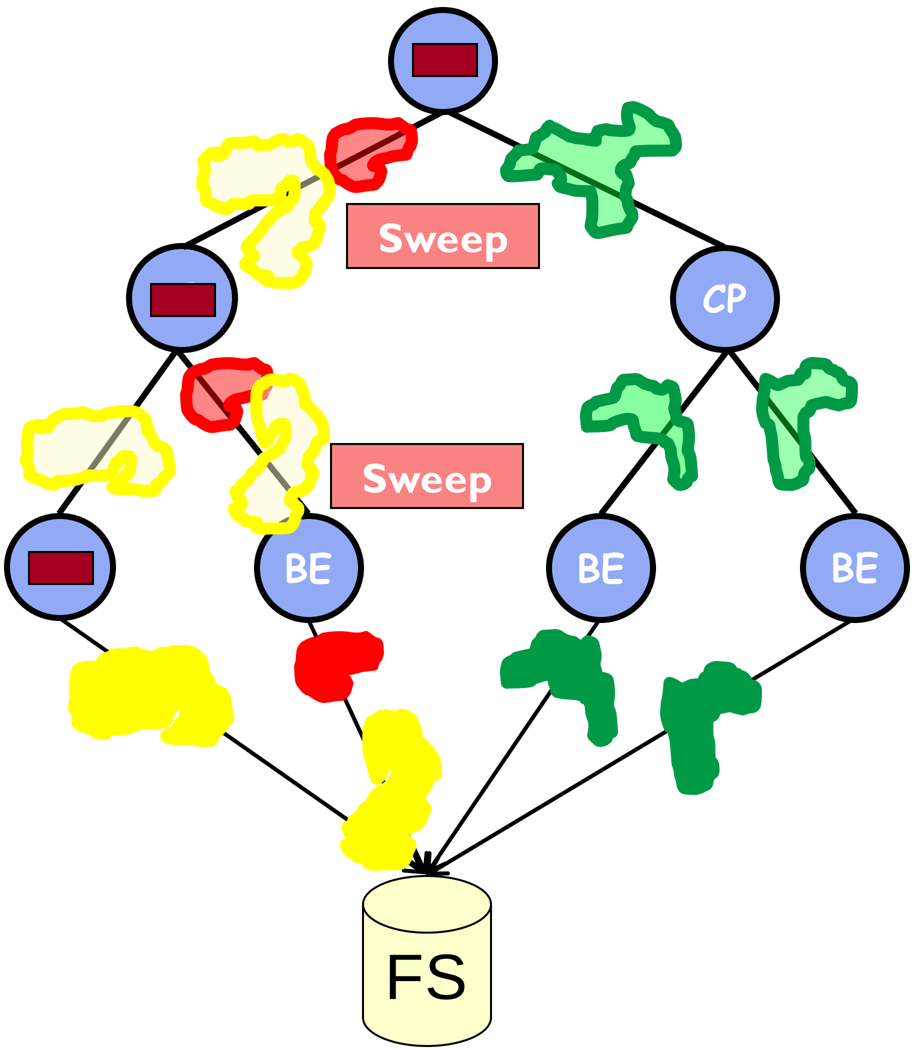
Partition: Distributed

DBSCAN: GPU (on BE)

Merge: CPU (x #levels)

Sweep: CPU (x #levels)

Intro to Mr. Scan



Mr. Scan Phases

Partition: Distributed

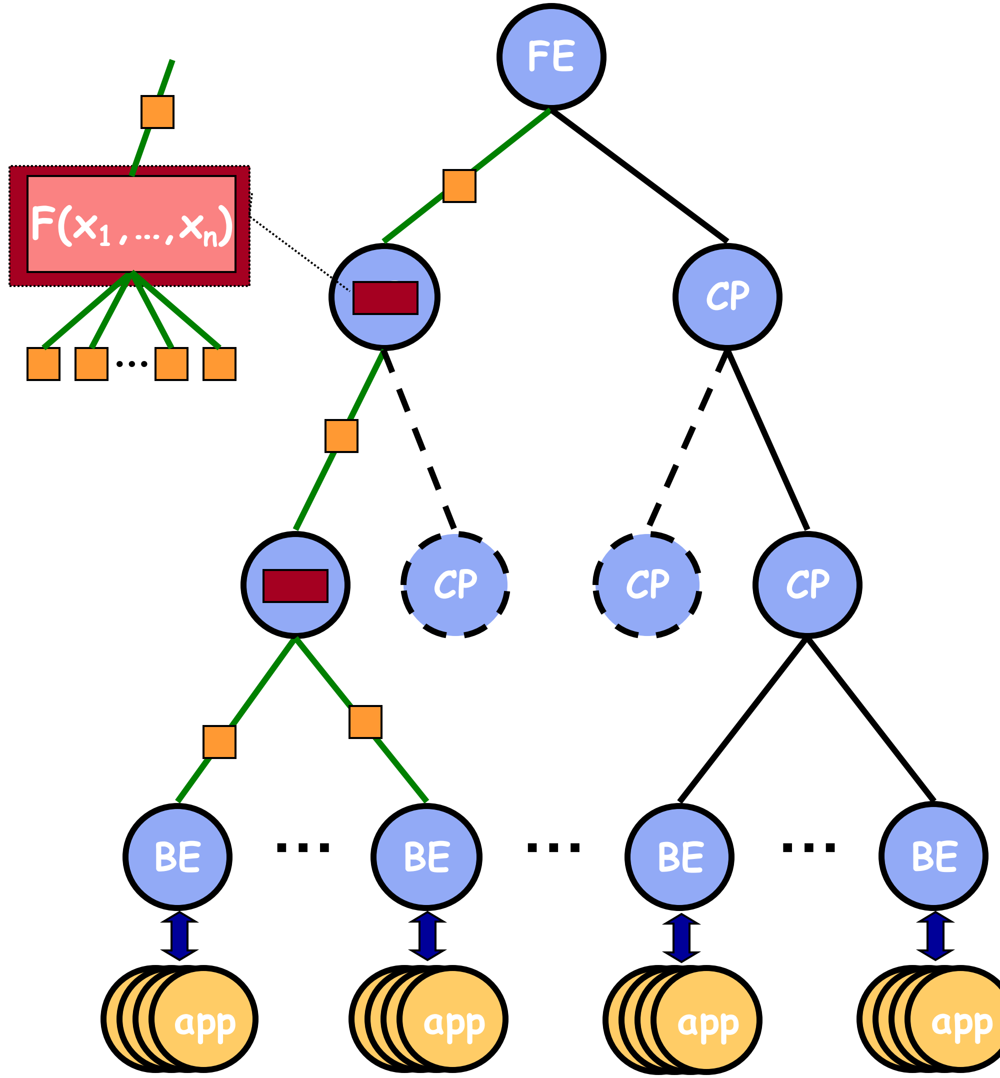
DBSCAN: GPU (on BE)

Merge: CPU (x #levels)

Sweep: CPU (x #levels)

Properties of a Scalable TBON App

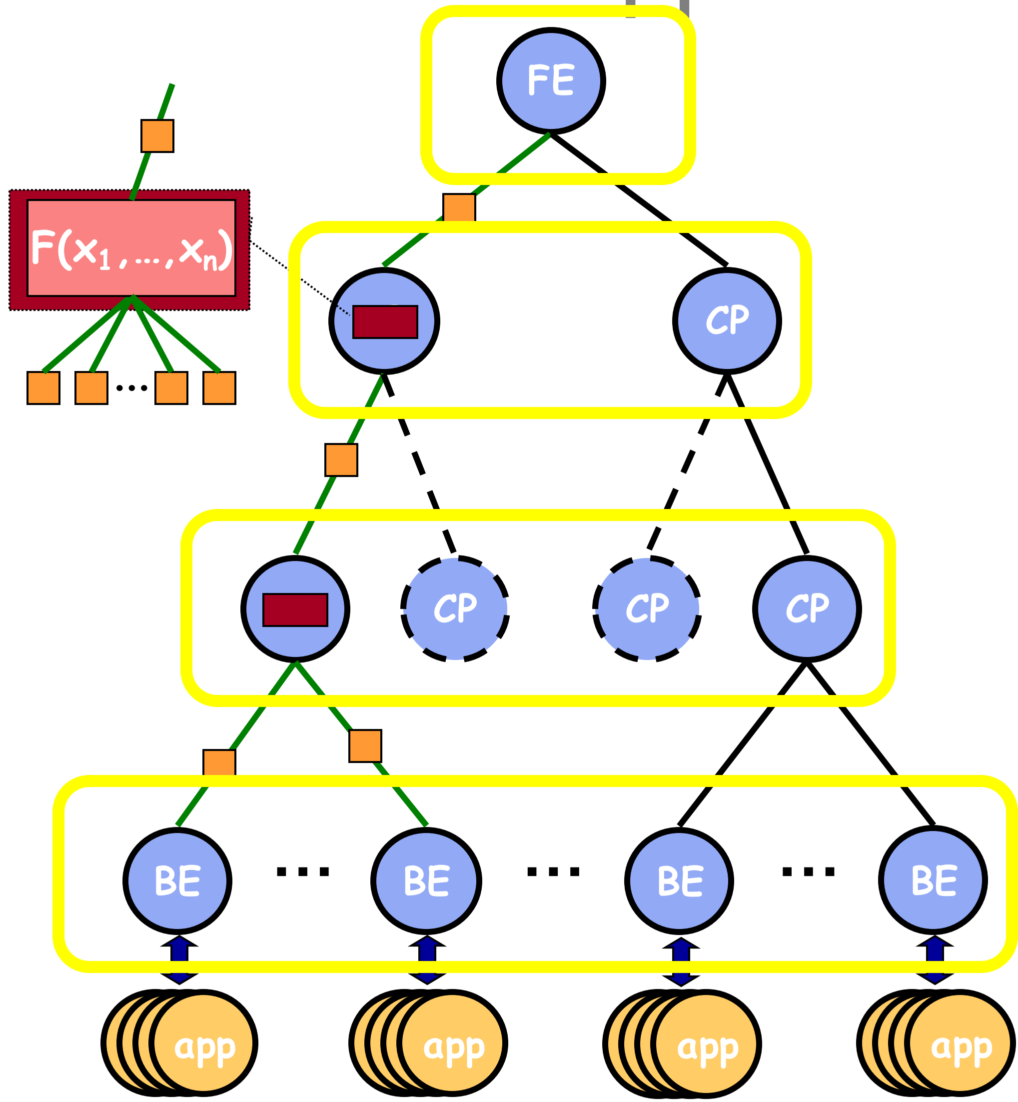
Applications must have the following properties to be scalable in a TBON:



Properties of a Scalable TBON App

Applications must have the following properties to be scalable in a TBON:

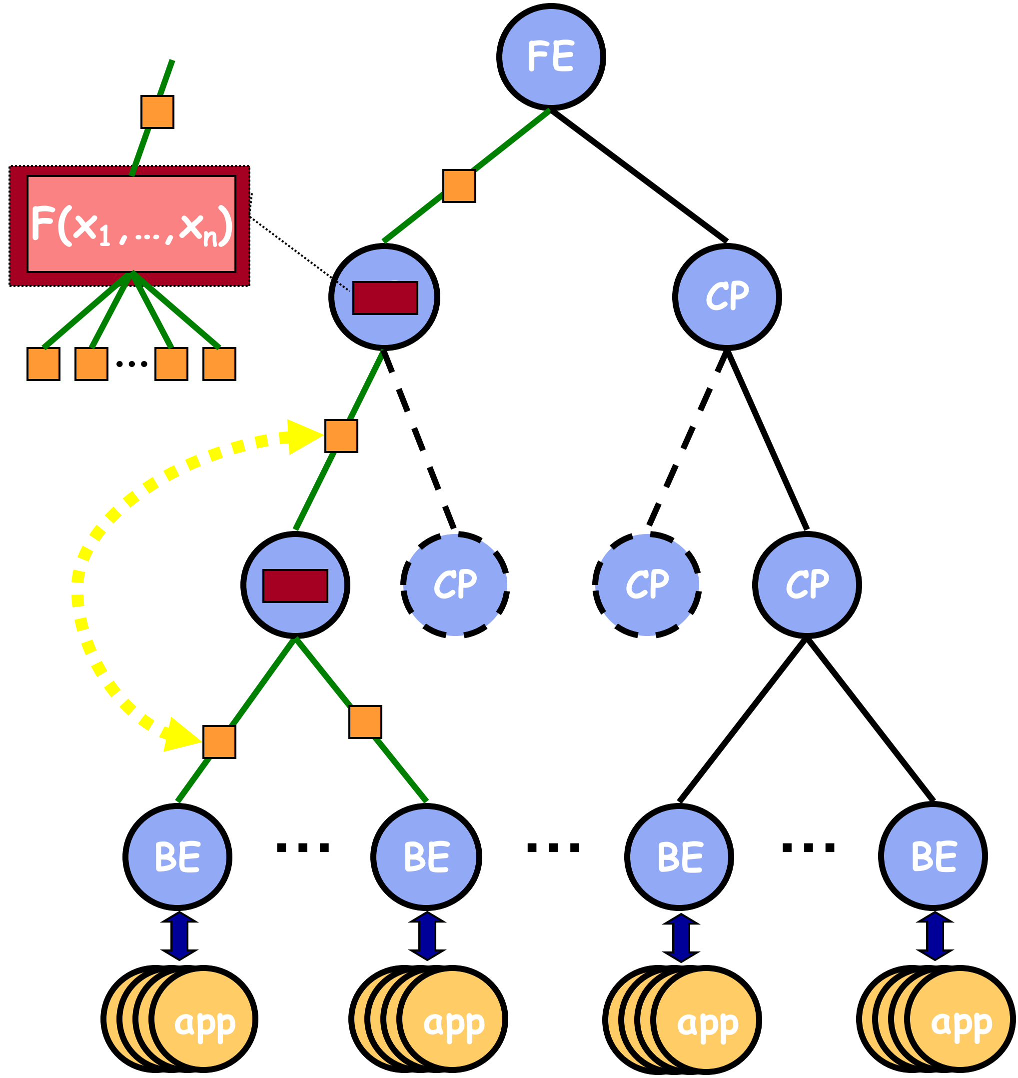
- Same amount of work across all nodes in the tree.



Properties of a Scalable TBON App

Applications must have the following properties to be scalable in a TBON:

- Same amount of work across all nodes in the tree.
- Size of reduction output must be equal or less than input size.

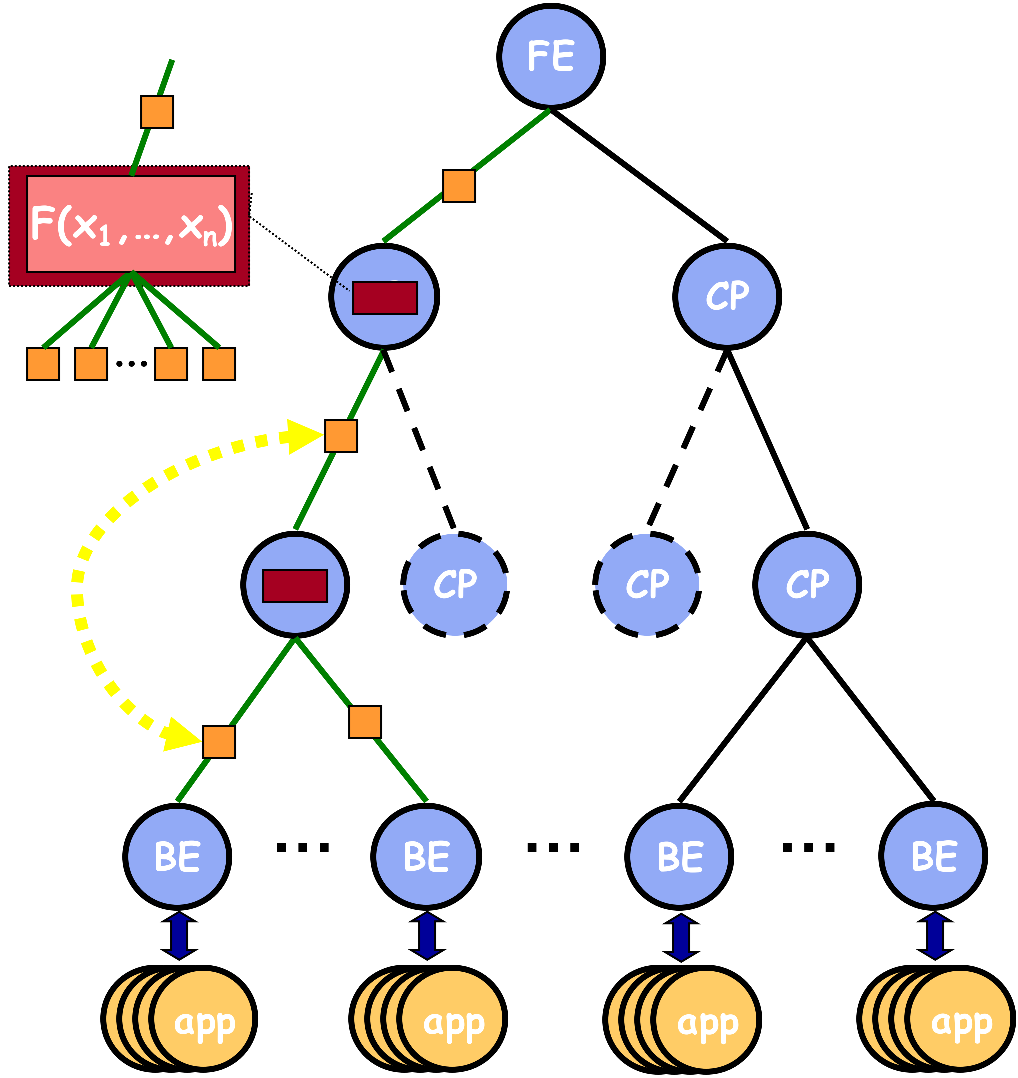


Properties of a Scalable TBON App

Applications must have the following properties to be scalable in a TBON:

- Same amount of work across all nodes in the tree.
- Size of reduction output must be equal or less than input size.

Mr. Scan must have these properties to scale



Challenges To Scaling DBSCAN

Workload Balance:

- DBSCAN requires that points near one another must be processed on the same node.

Size of data needed for merging:

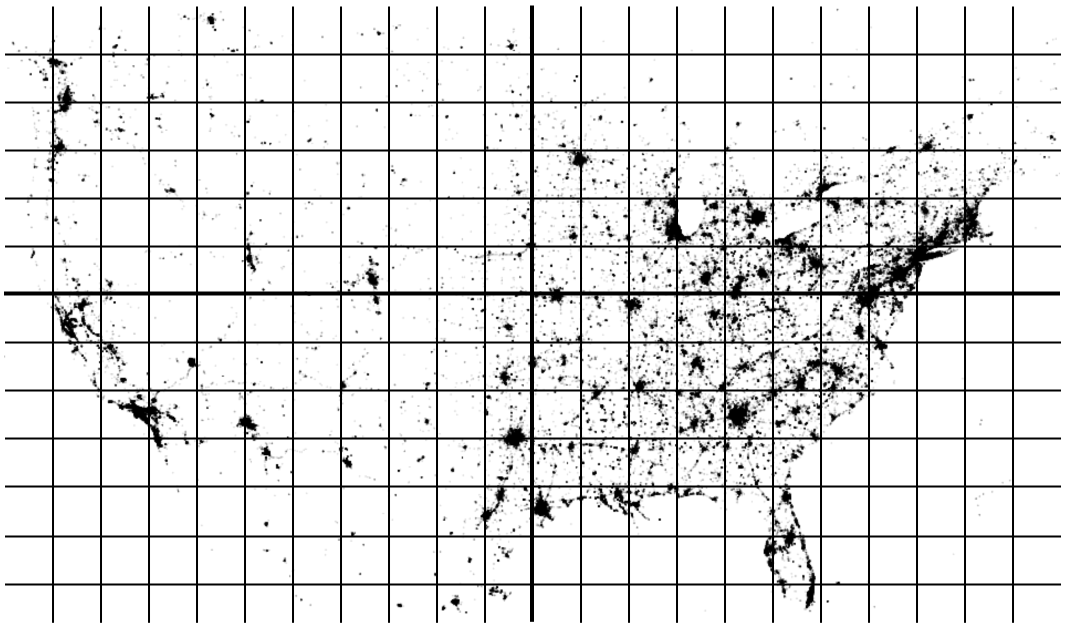
- Simple merging methods require sending all points in all clusters to merge accurately
  - Sending billions of points in the merge phase is infeasible.

Our Solutions

Solve the balance and data size issues jointly with the following techniques:

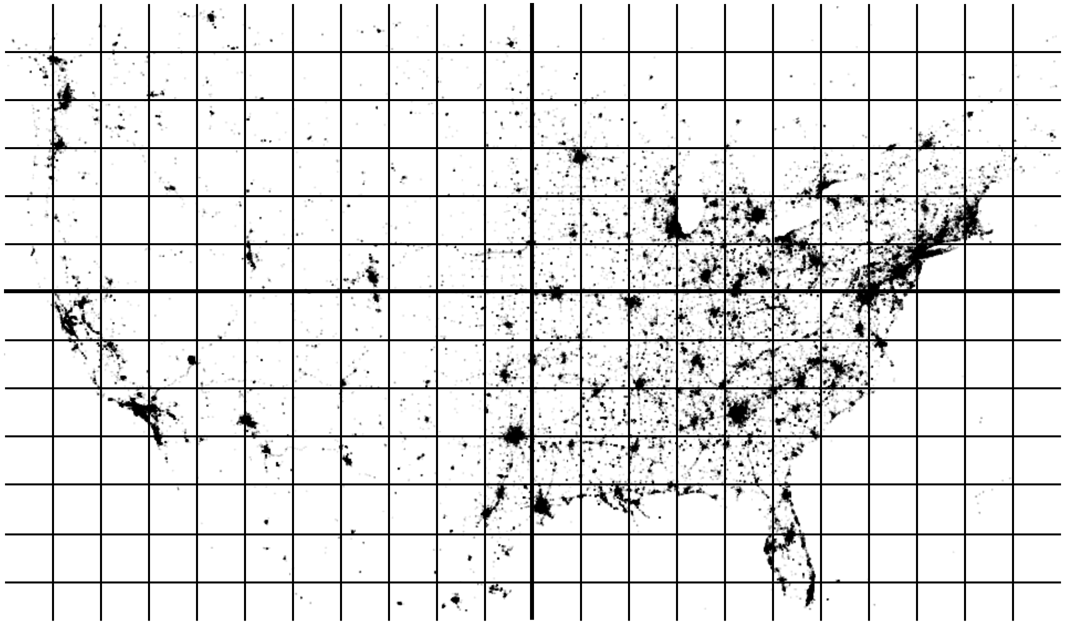
- Smart Partitioning
  - Selectively performing redundant computation to reduce merge data size (using data duplicated at edge of each partition).
  - Use an estimate of partition density to identify hard to compute partitions.
- Dense Box Algorithm
  - Reduces the complexity of computation of extremely dense partitions.
- Representative Points
  - Leverage the redundant computation to create a merge method that requires only a small fixed subset of points.

Partition Phase



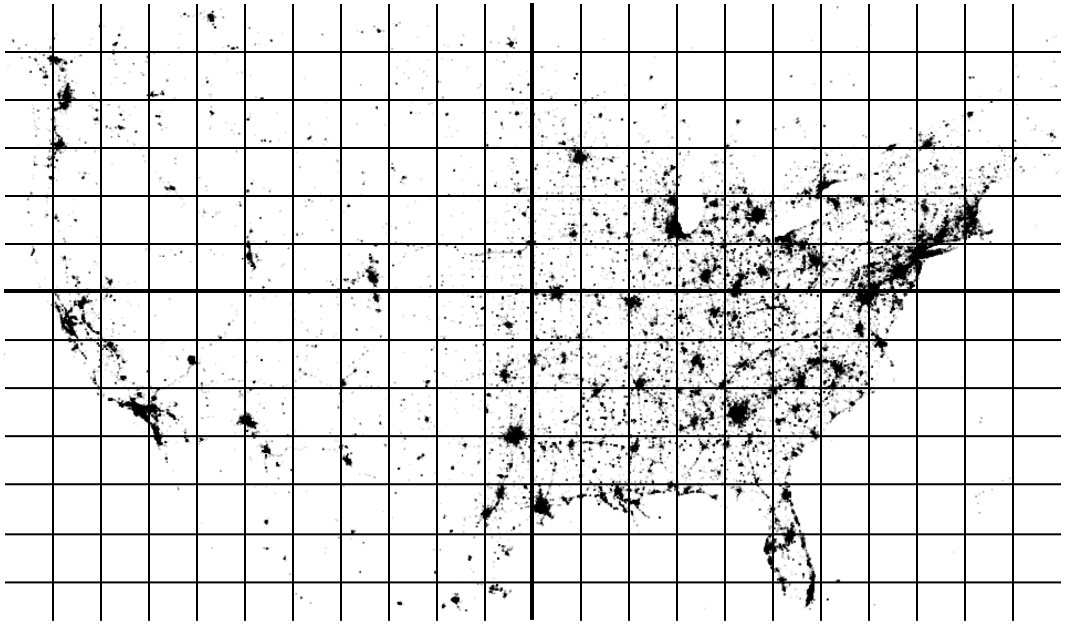
Partition Phase

- Algorithm:



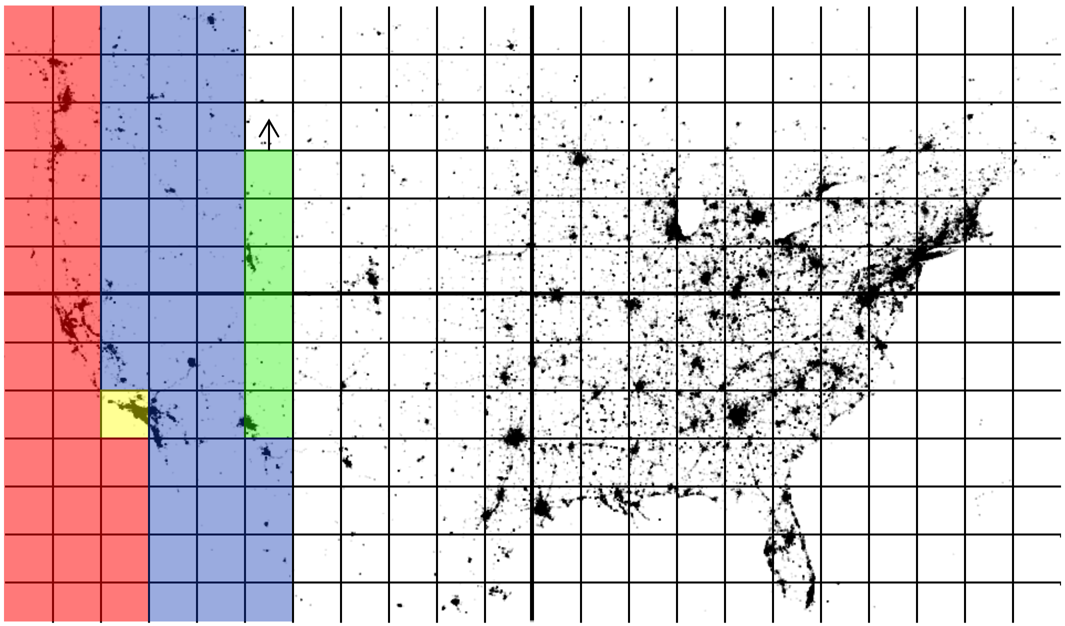
Partition Phase

- Algorithm:
  - Form initial partitions



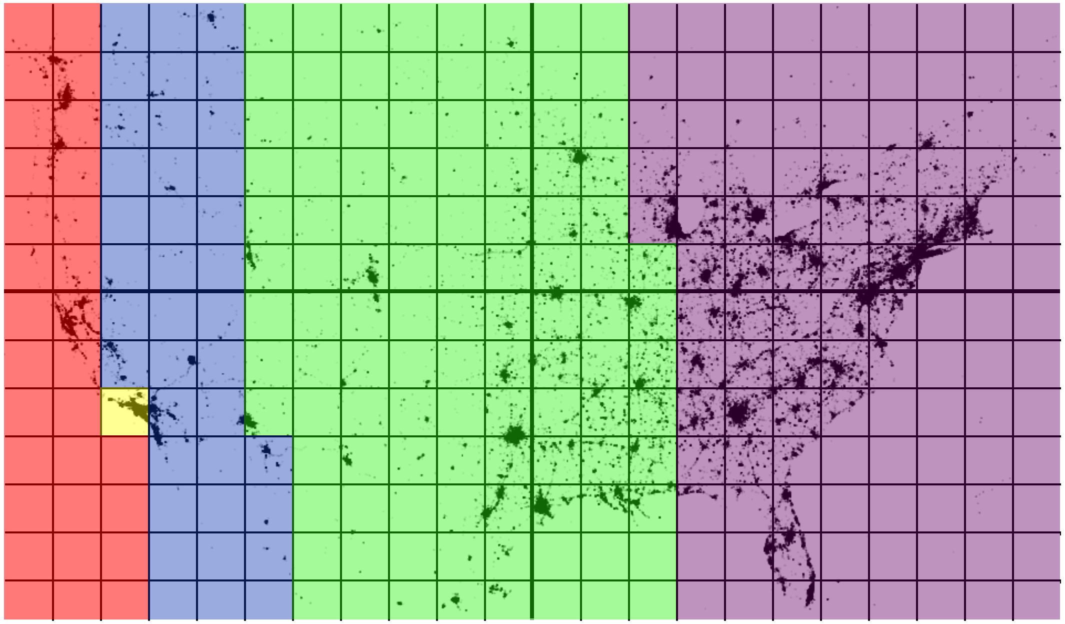
Partition Phase

- Algorithm:
  - Form initial partitions



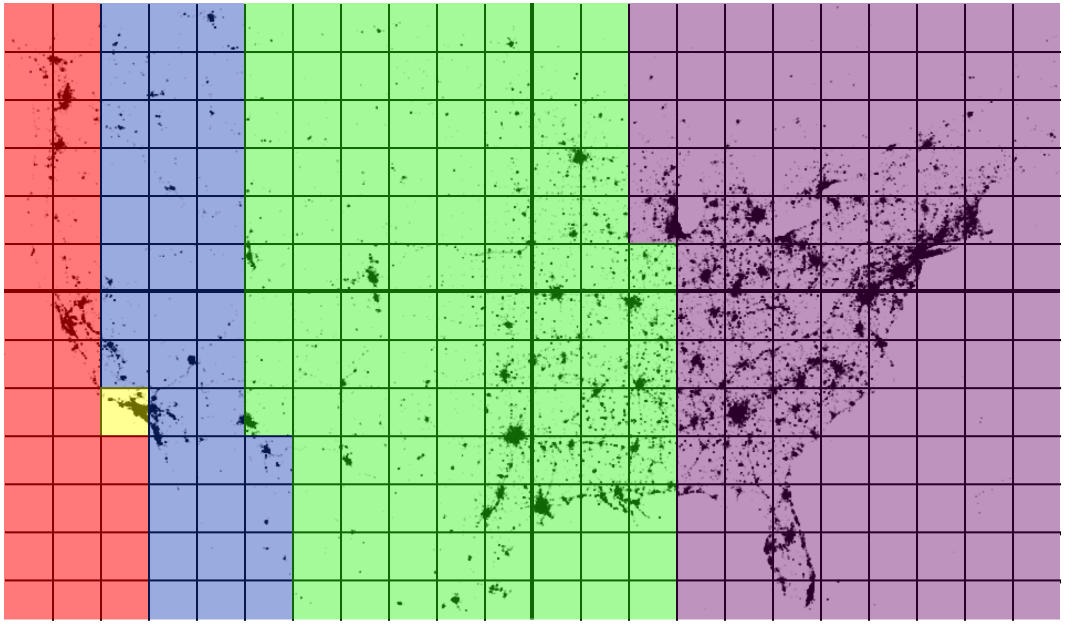
Partition Phase

- Algorithm:
  - Form initial partitions



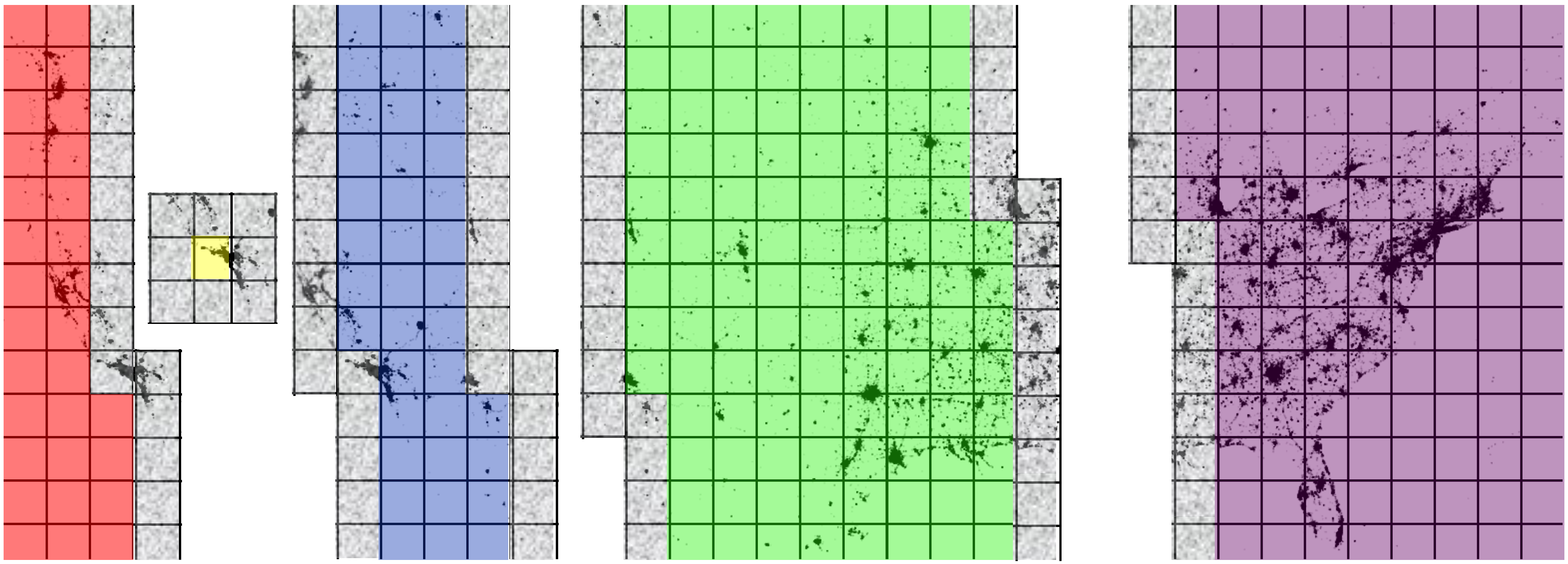
Partition Phase

- Algorithm:
  - Form initial partitions
  - Add shadow regions



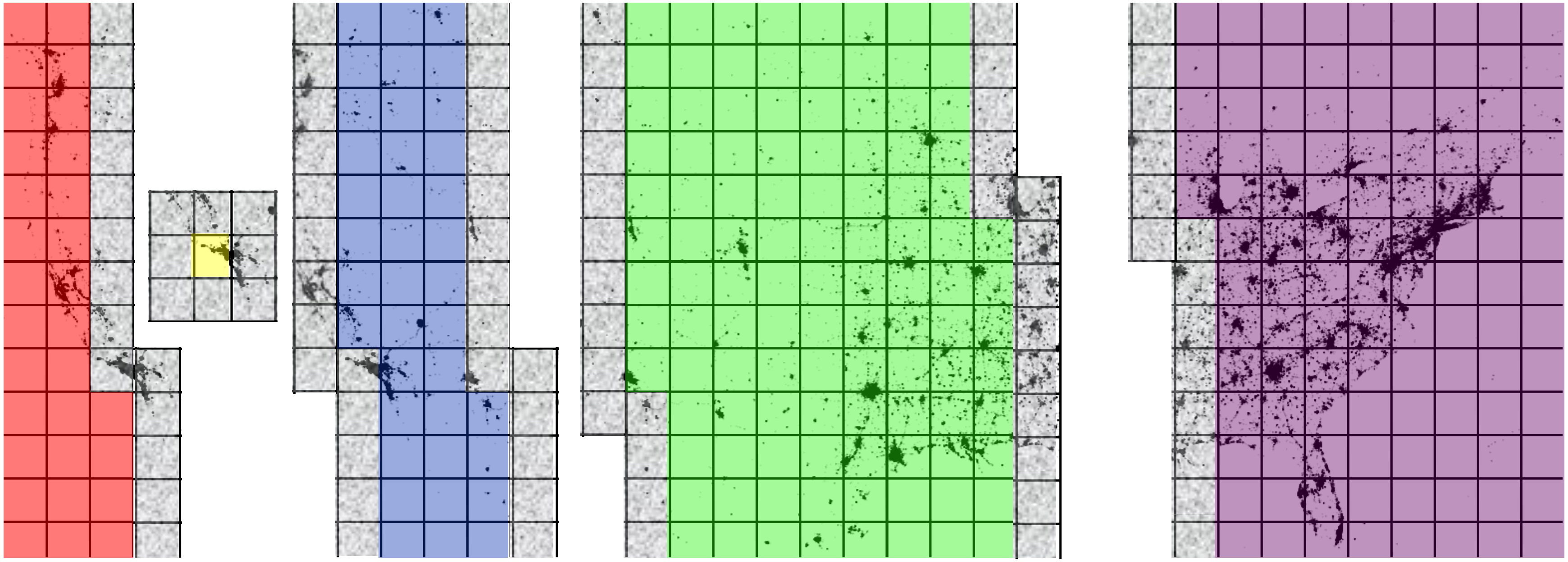
Partition Phase

- Algorithm:
  - Form initial partitions
  - Add shadow regions



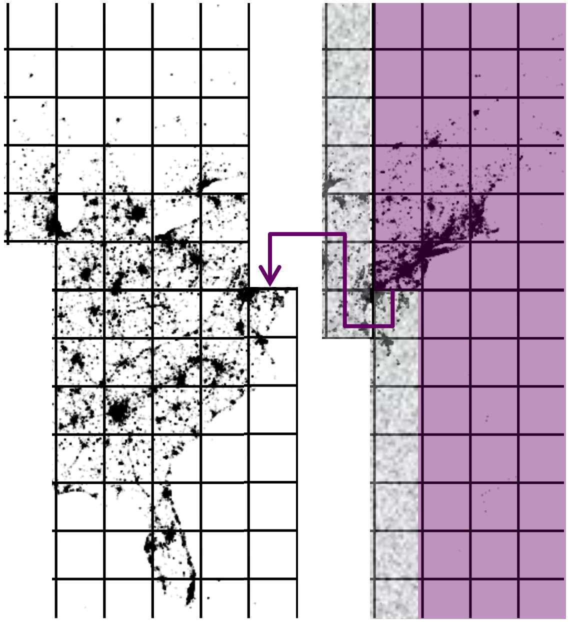
Partition Phase

- Algorithm:
  - Form initial partitions
  - Add shadow regions
  - Rebalance



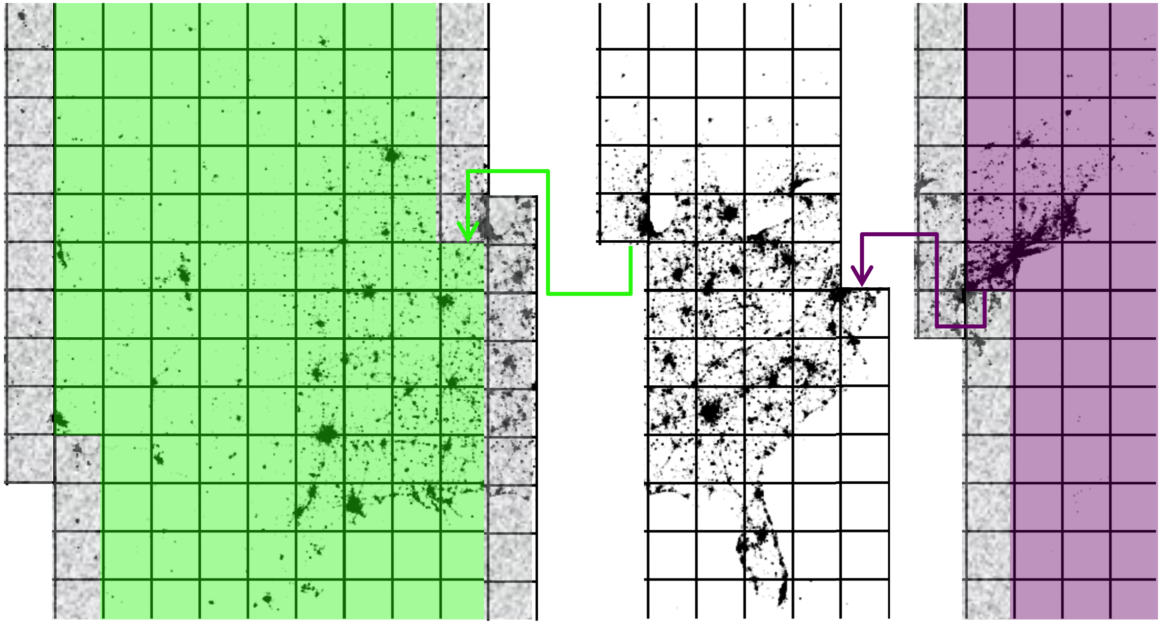
Partition Phase

- Algorithm:
  - Form initial partitions
  - Add shadow regions
  - Rebalance



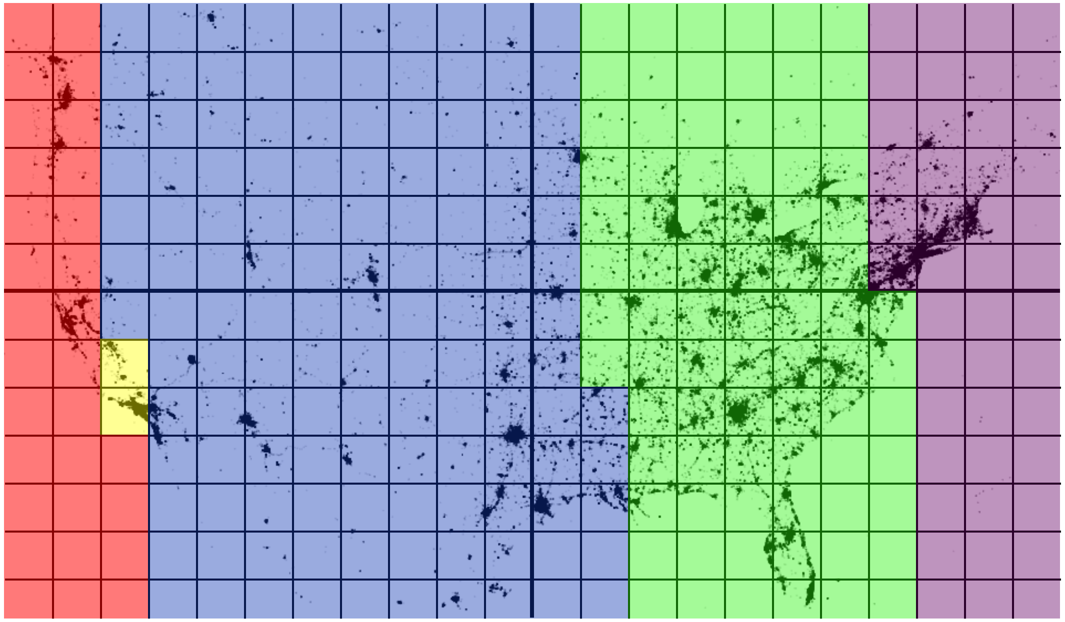
Partition Phase

- Algorithm:
  - Form initial partitions
  - Add shadow regions
  - Rebalance



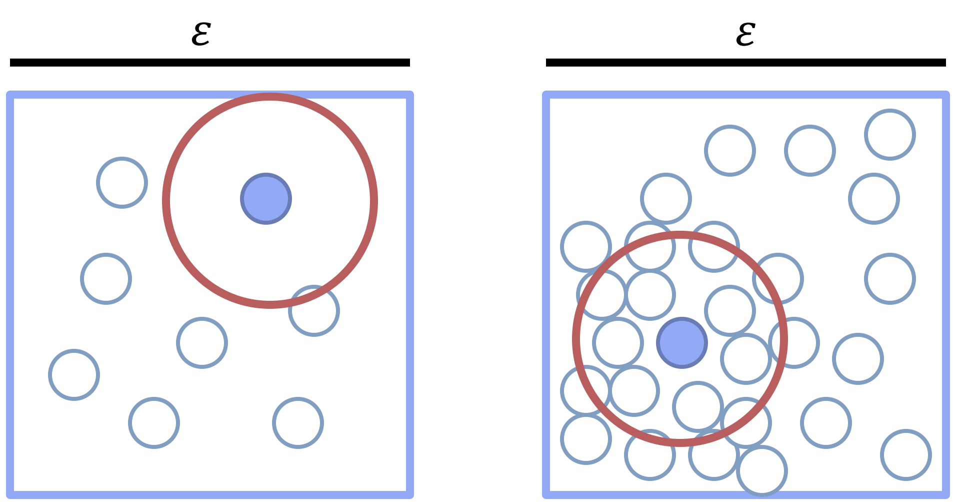
Partition Phase

- Algorithm:
  - Form initial partitions
  - Add shadow regions
  - Rebalance



The DBSCAN Density Problem

- Imbalances in point density can cause huge differences in runtimes between Thread Groups inside of a GPU (10-15x variance in time)
  - Issue is caused by the lookup operation for a points neighbors in the DBSCAN point expansion phase.



Higher density results in higher neighbor count which increases the number of comparison operations

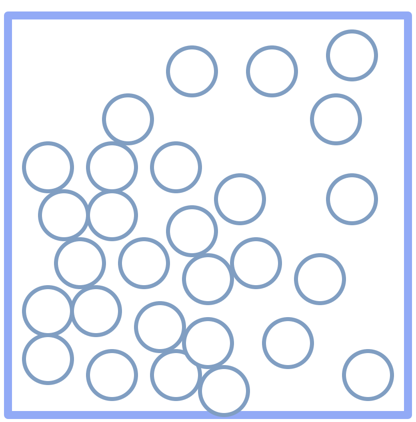
Dense Box

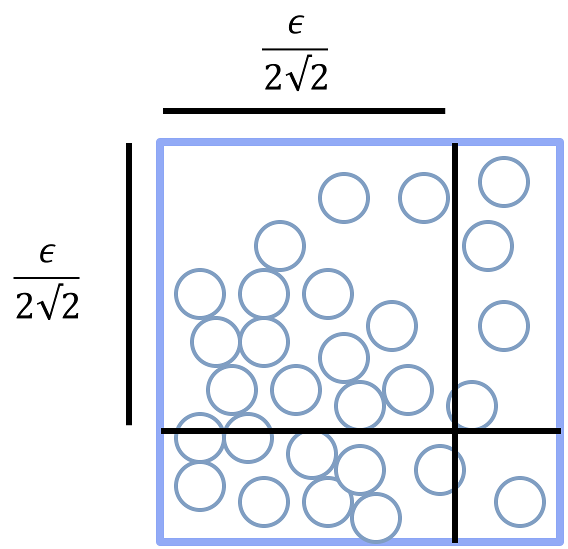
- Dense Box eliminates the need to perform neighbor lookups on points in dense regions by labeling points as a member of cluster before DBSCAN is run.

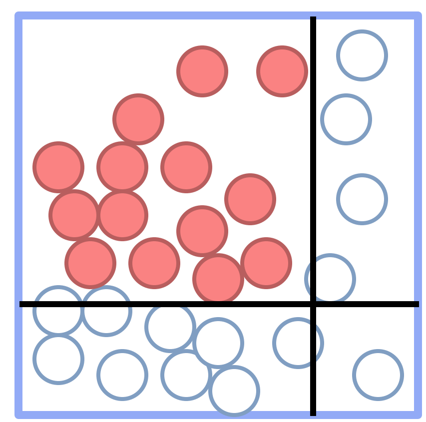
1. Start with an ε region.

2. Divide the region of data into area's of size $\frac{\epsilon}{2\sqrt{2}}$ for dense area detection*.

3. For each $\frac{\epsilon}{2\sqrt{2}}$ area which has point count >= MinPts. Mark points as members of a cluster. Do not expand these points.





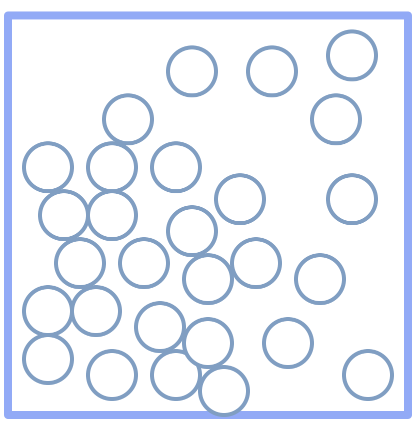


* $\frac{\epsilon}{2\sqrt{2}}$ chosen because it guarantees all points inside are within ϵ distance of each other.

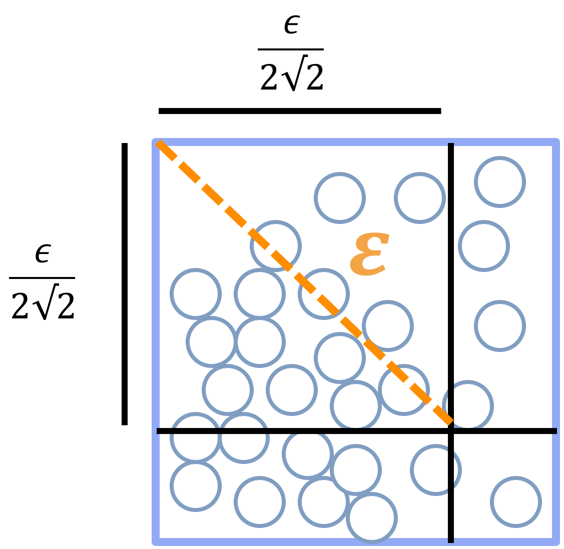
Dense Box

- Dense Box eliminates the need to perform neighbor lookups on points in dense regions by labeling points as a member of cluster before DBSCAN is run.

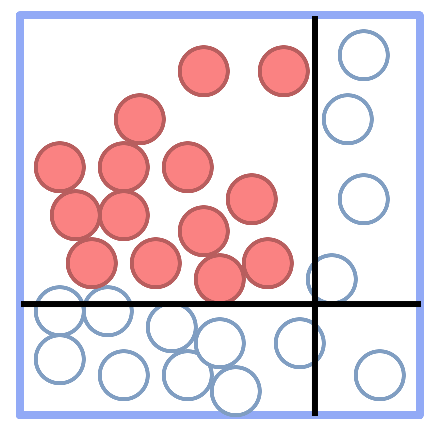
1. Start with an ε region.



2. Divide the region of data into area's of size $\frac{\epsilon}{2\sqrt{2}}$ for dense area detection*.



3. For each $\frac{\epsilon}{2\sqrt{2}}$ area which has point count >= MinPts. Mark points as members of a cluster. Do not expand these points.

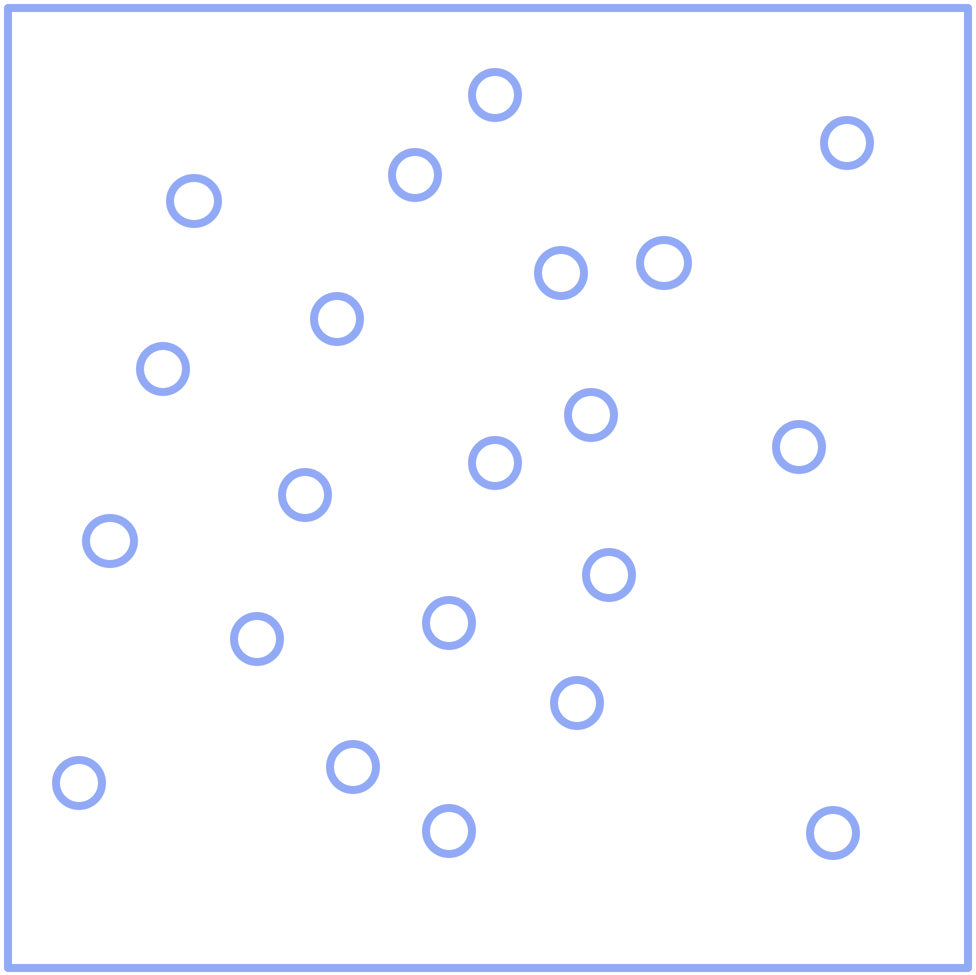


* $\frac{\epsilon}{2\sqrt{2}}$ chosen because it guarantees all points inside are within ϵ distance of each other.

Merge Algorithm

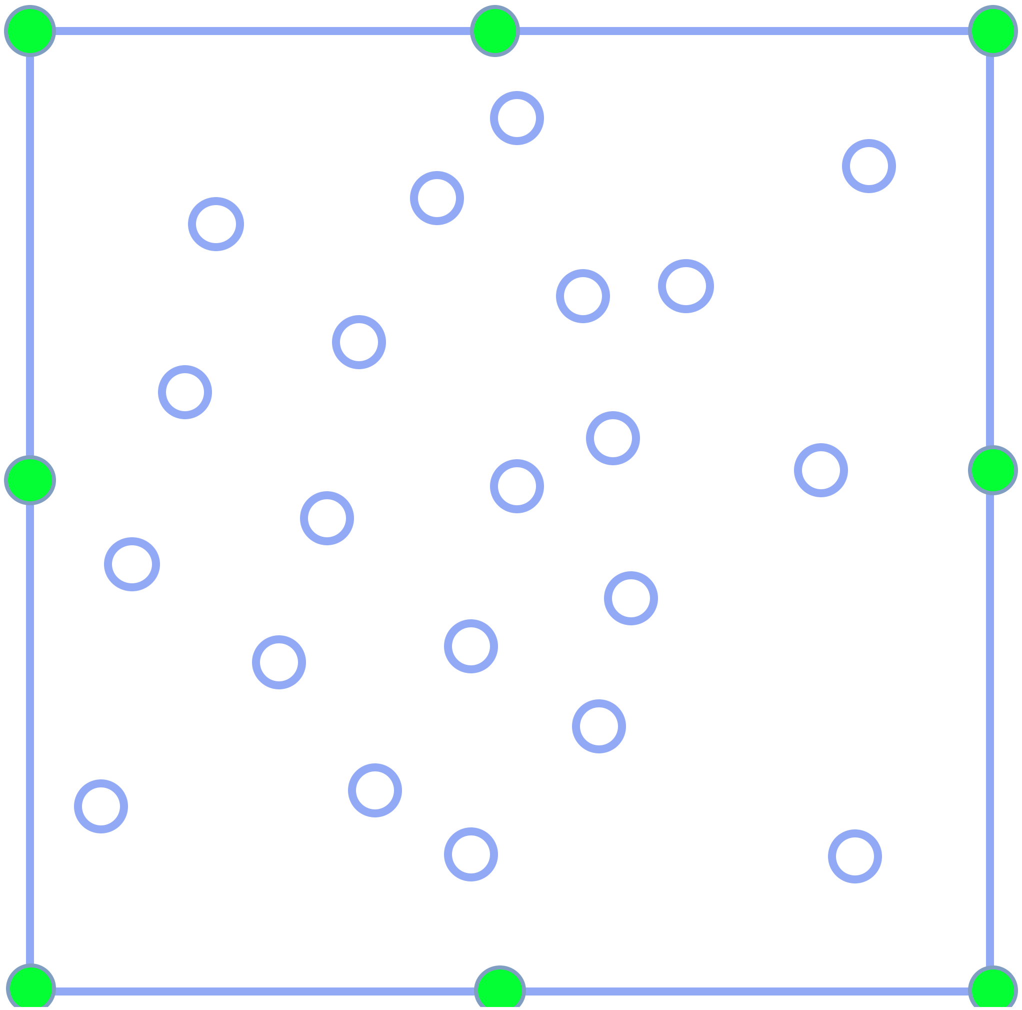
- Two phases in the merge operation
  1. Select Representative points (Leaf Node)
  2. Merge operation (Internal Node)

Representative Points (Leaf Nodes)



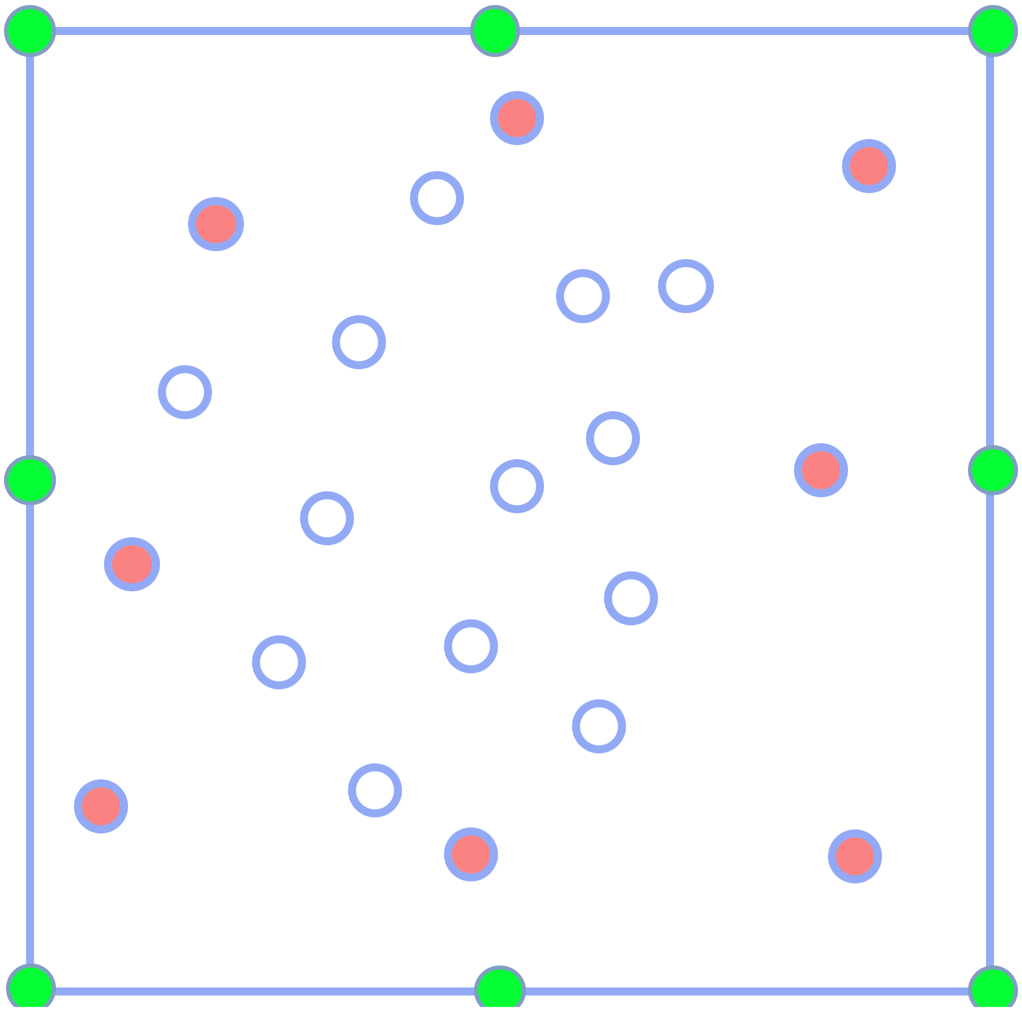
- Large clusters are too expensive to move up the tree
- In border regions we select eight representative points to represent the cluster
- These points guarantee that any overlap of clusters detected on adjacent nodes

Representative Points (Leaf Nodes)



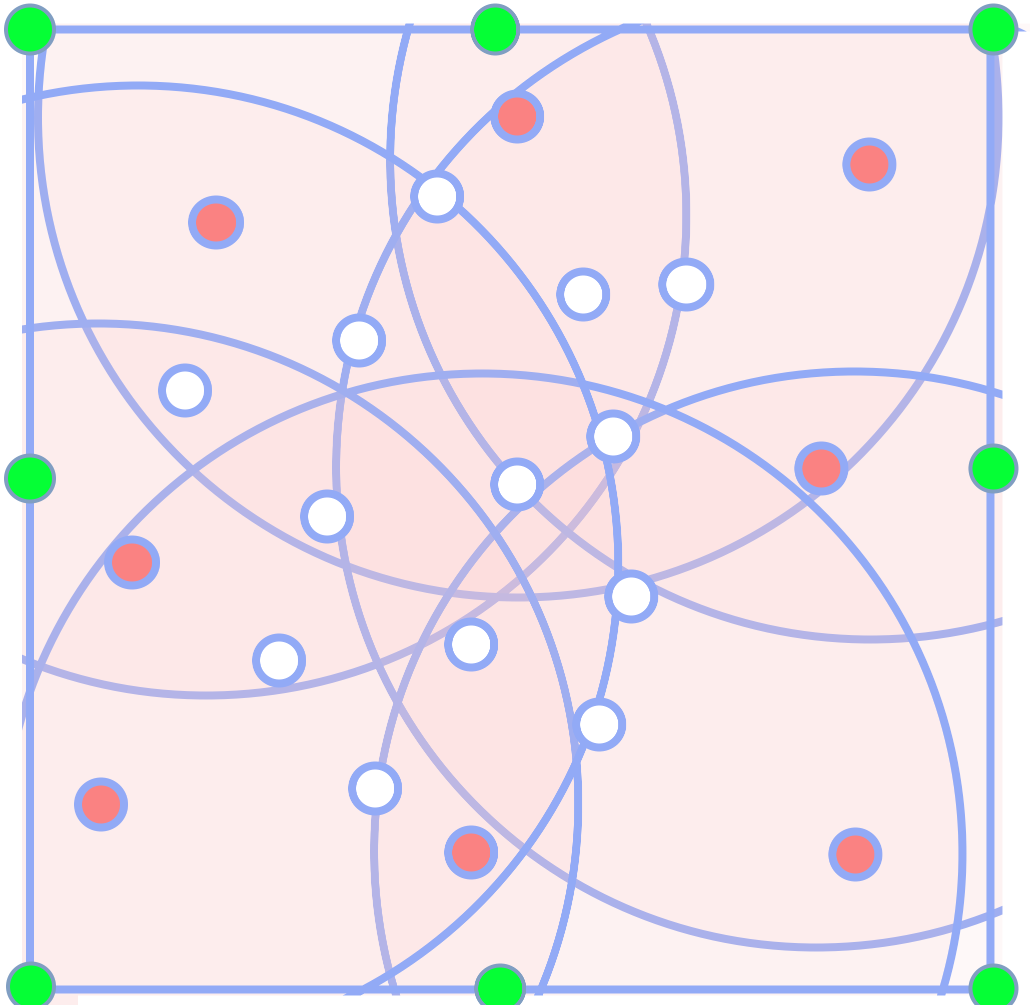
- Large clusters are too expensive to move up the tree
- In border regions we select eight representative points to represent the cluster
- These points guarantee that any overlap of clusters detected on adjacent nodes

Representative Points (Leaf Nodes)



- Large clusters are too expensive to move up the tree
- In border regions we select eight representative points to represent the cluster
- These points guarantee that any overlap of clusters detected on adjacent nodes

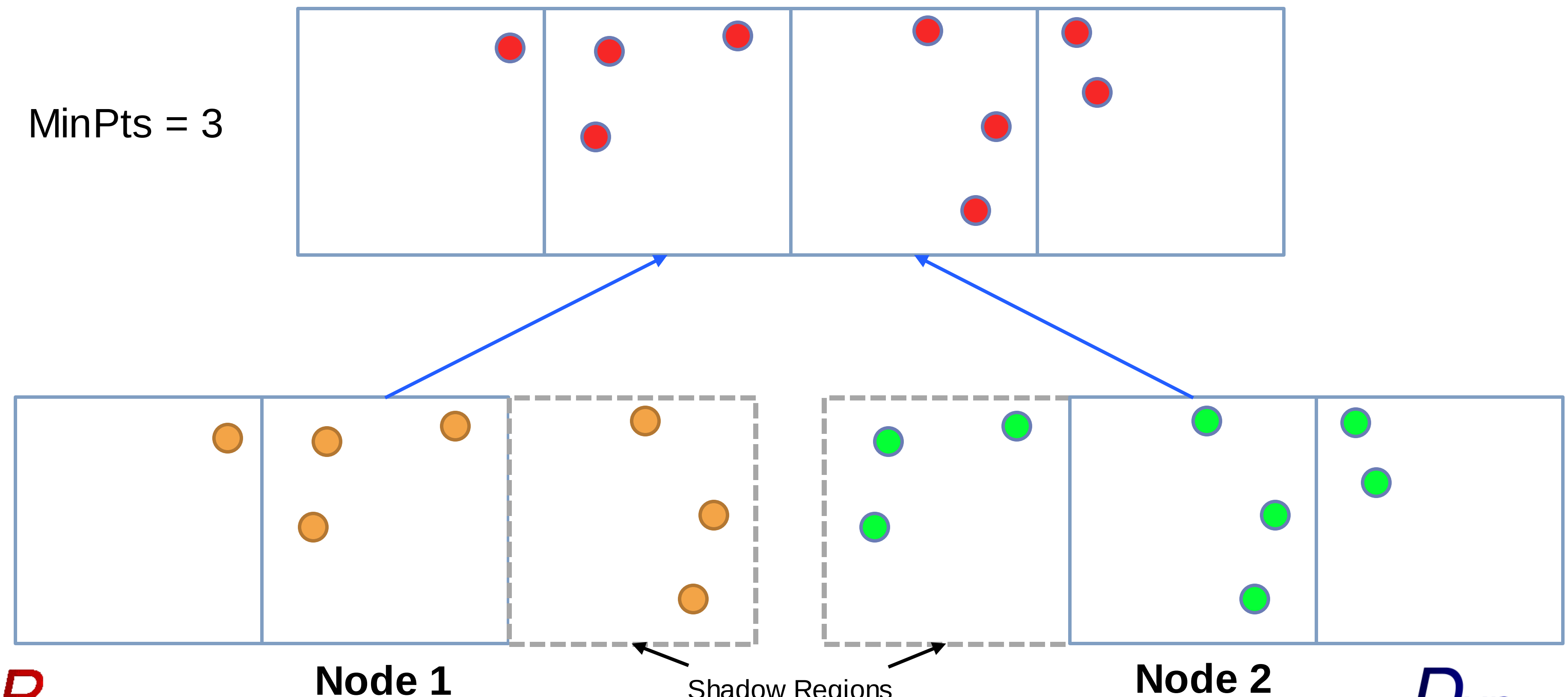
Representative Points (Leaf Nodes)



- Large clusters are too expensive to move up the tree
- In border regions we select eight representative points to represent the cluster
- These points guarantee that any overlap of clusters detected on adjacent nodes

Merge Operation (Internal Nodes)

- Merge overlapping clusters found on different DBSCAN leaf nodes
- Merge needs to have low overhead and must operate without entire dataset (Representative Points + Non-Core points)

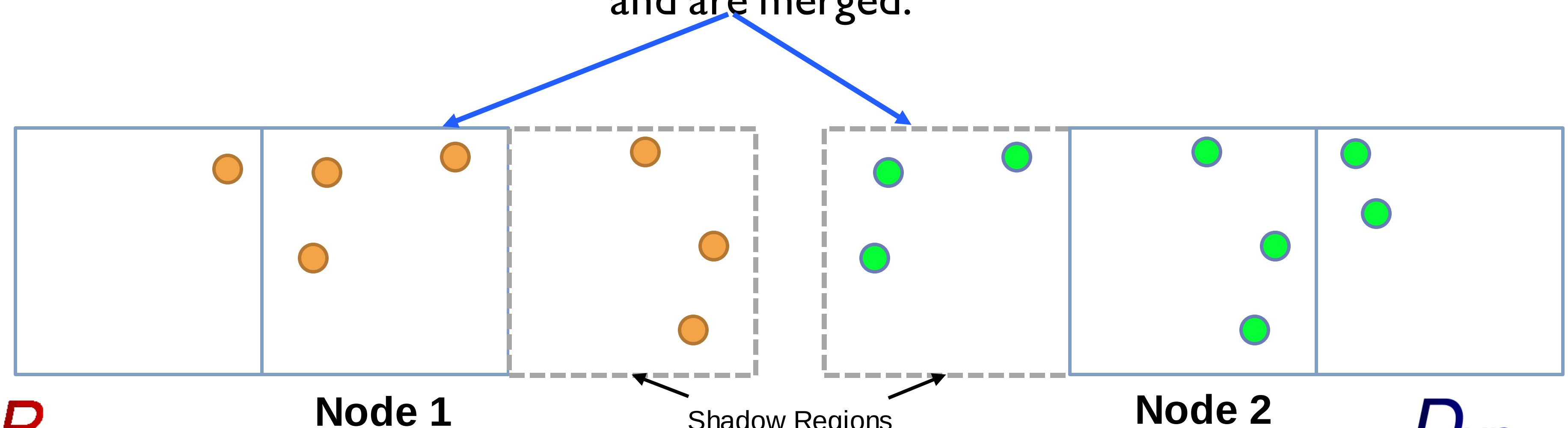


Merge Operation (Internal Nodes)

Compare representative points sent by leaf nodes

- If an overlap in representative points exists between two nodes, those clusters merge.
- Otherwise, the clusters are complete and no further propagation of representative points occurs.

Cluster on Node 1 and 2 have overlapping representative points and are merged.

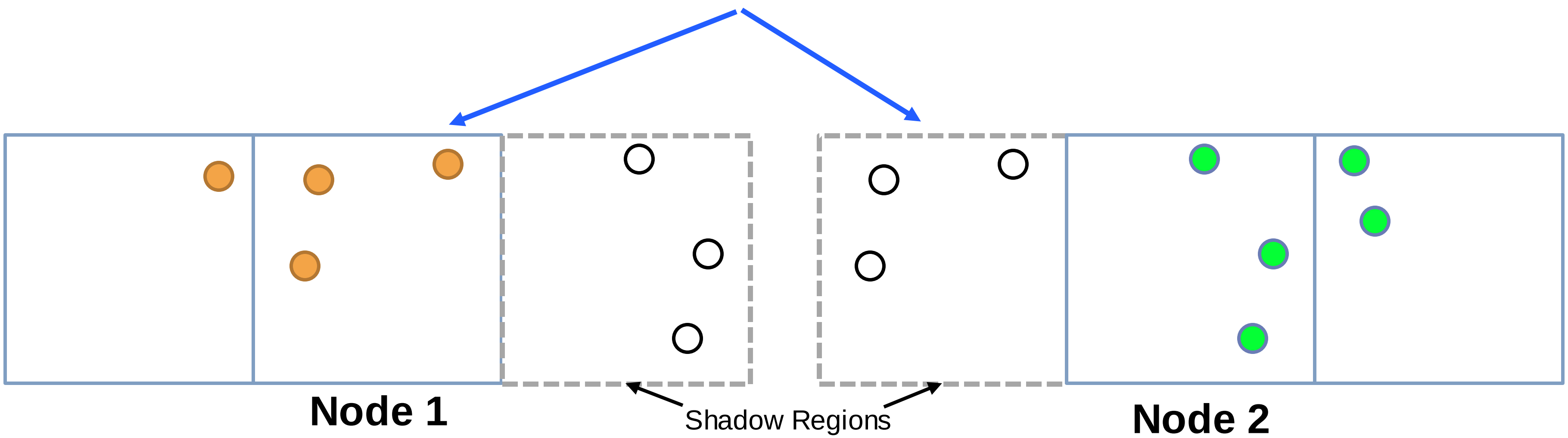


Merge Operation (Internal Nodes)

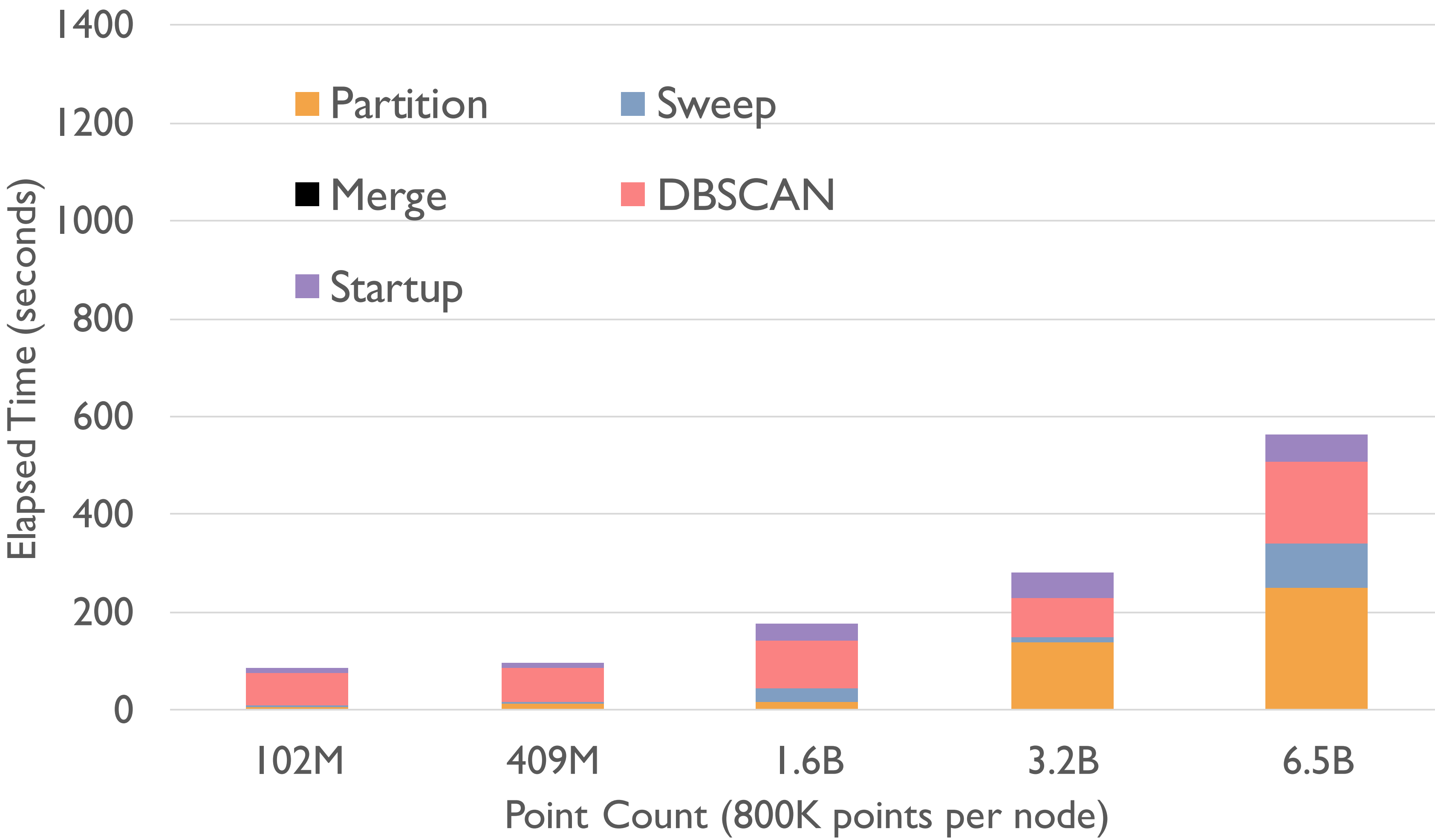
Compare representative points sent by leaf nodes

- If no overlap exists between clusters, finalize the clusters and do not propagate representative points.

No overlap in regions between Node 1 and 2. Clusters in both nodes are now final and no further propagation is needed



Mr. Scan Results



Takeaway Lessons for Scalable Data Reduction

- By selectively duplicating processing, we could use a less data intensive merging algorithm
- We were able to equalize the workload between nodes by use of the dense box algorithm
- Using these approaches, we were able to scale Mr. Scan up to 8192 nodes.
- MRNet is a general scalability framework for data reductions

Paper available at: http://paradyn.org

E-mail: welton@cs.wisc.edu

Questions?