Tree-Based Density Clustering using Graphics Processors

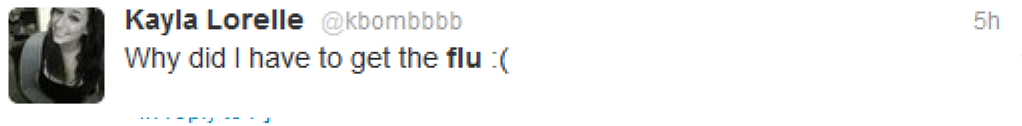
A First Marriage of MRNet and GPUs

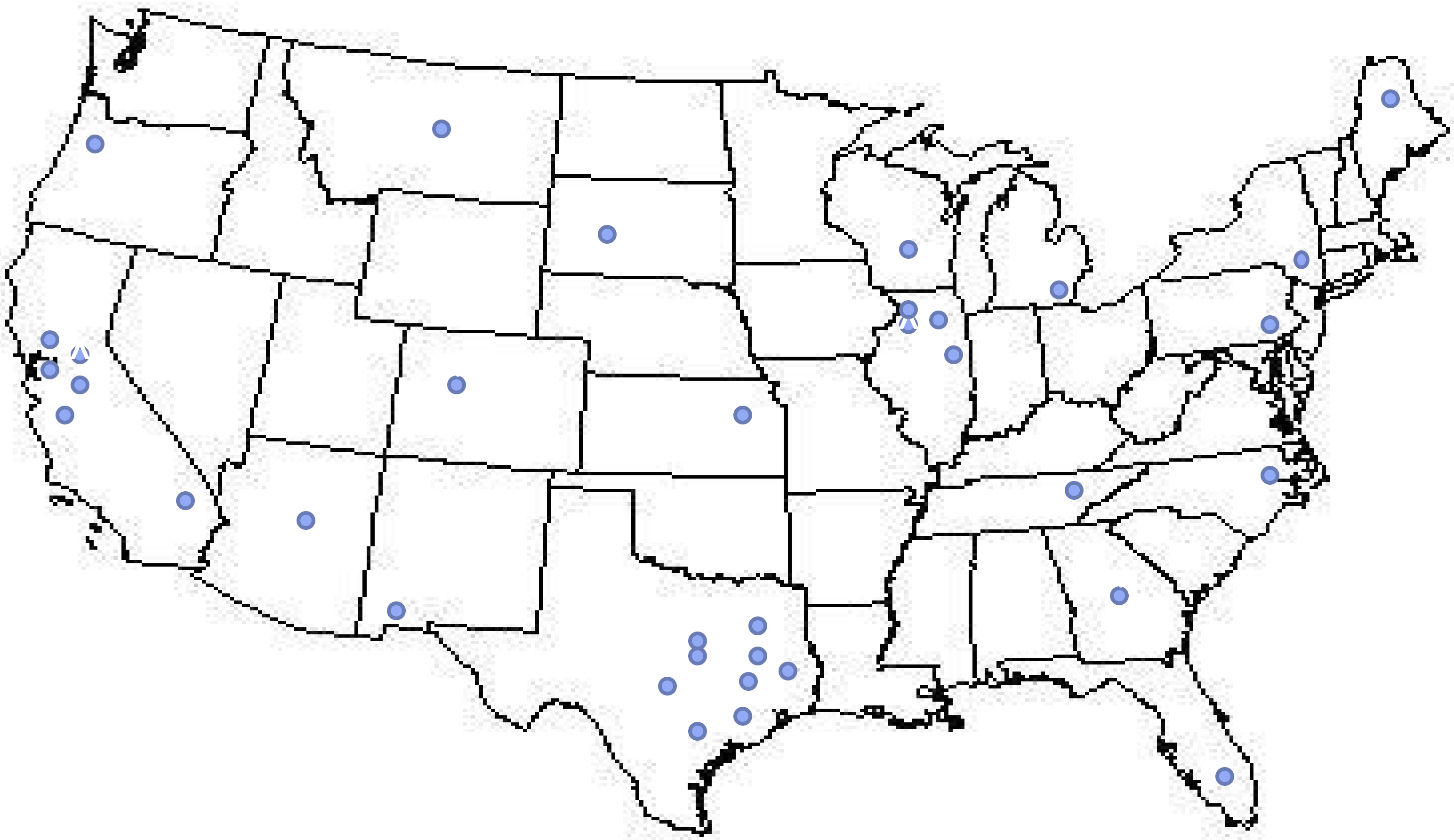
Evan Samanas and Ben Welton

Paradyn Project

Paradyn / Dyninst Week
College Park, Maryland
March 26-28, 2012

The Tweet Stream

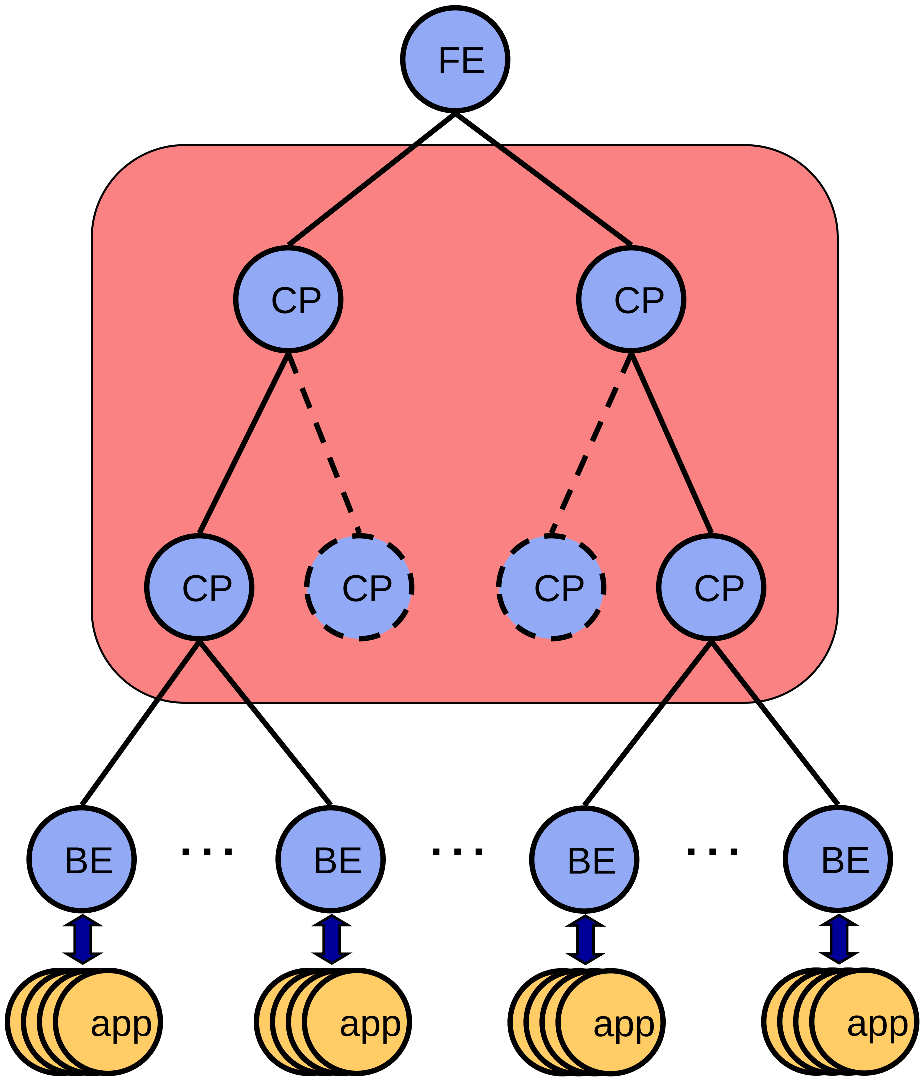




Source: Twitter, Map: About.com

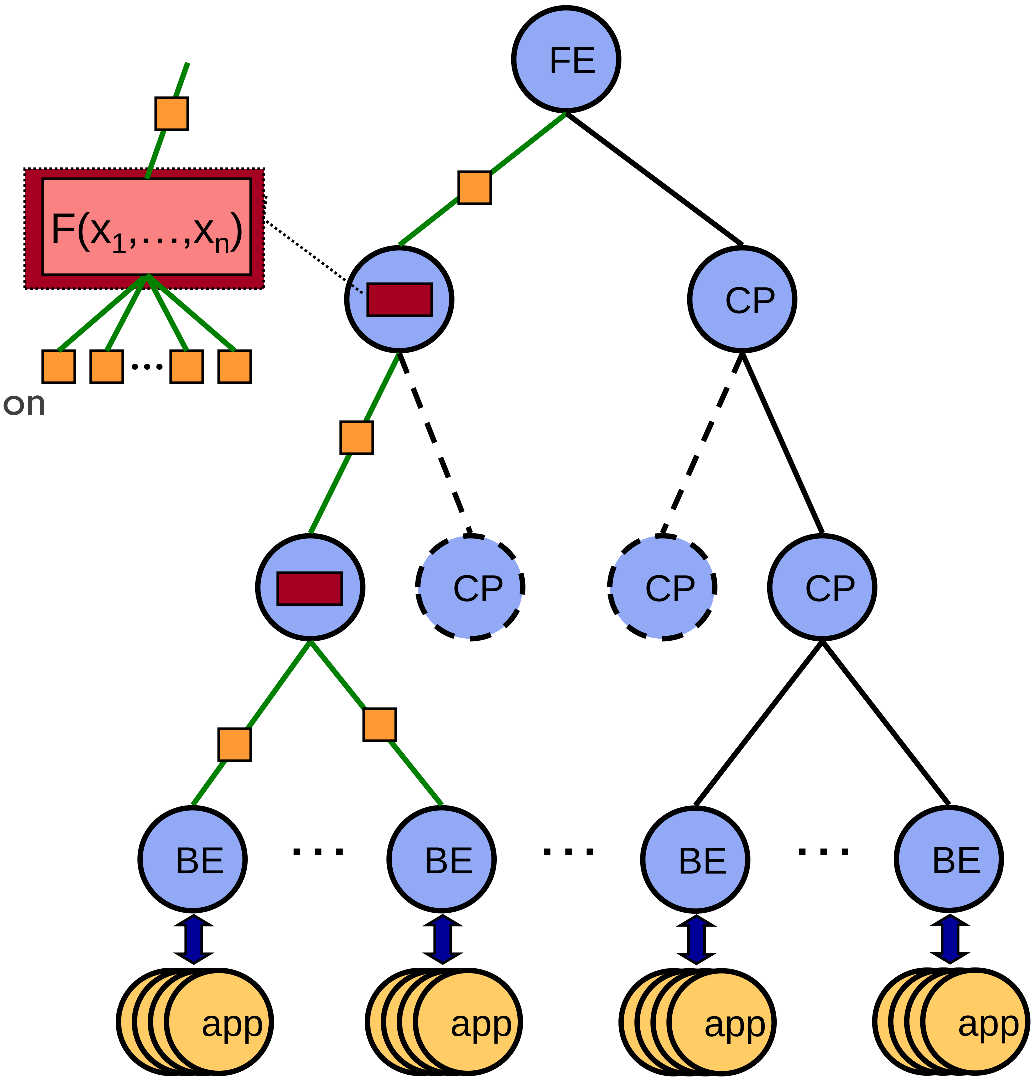
Tree-Based Overlay Networks (TBONs)

- Scalable multicast
- Scalable gather
- Scalable data aggregation



MRNet – Multicast / Reduction Network

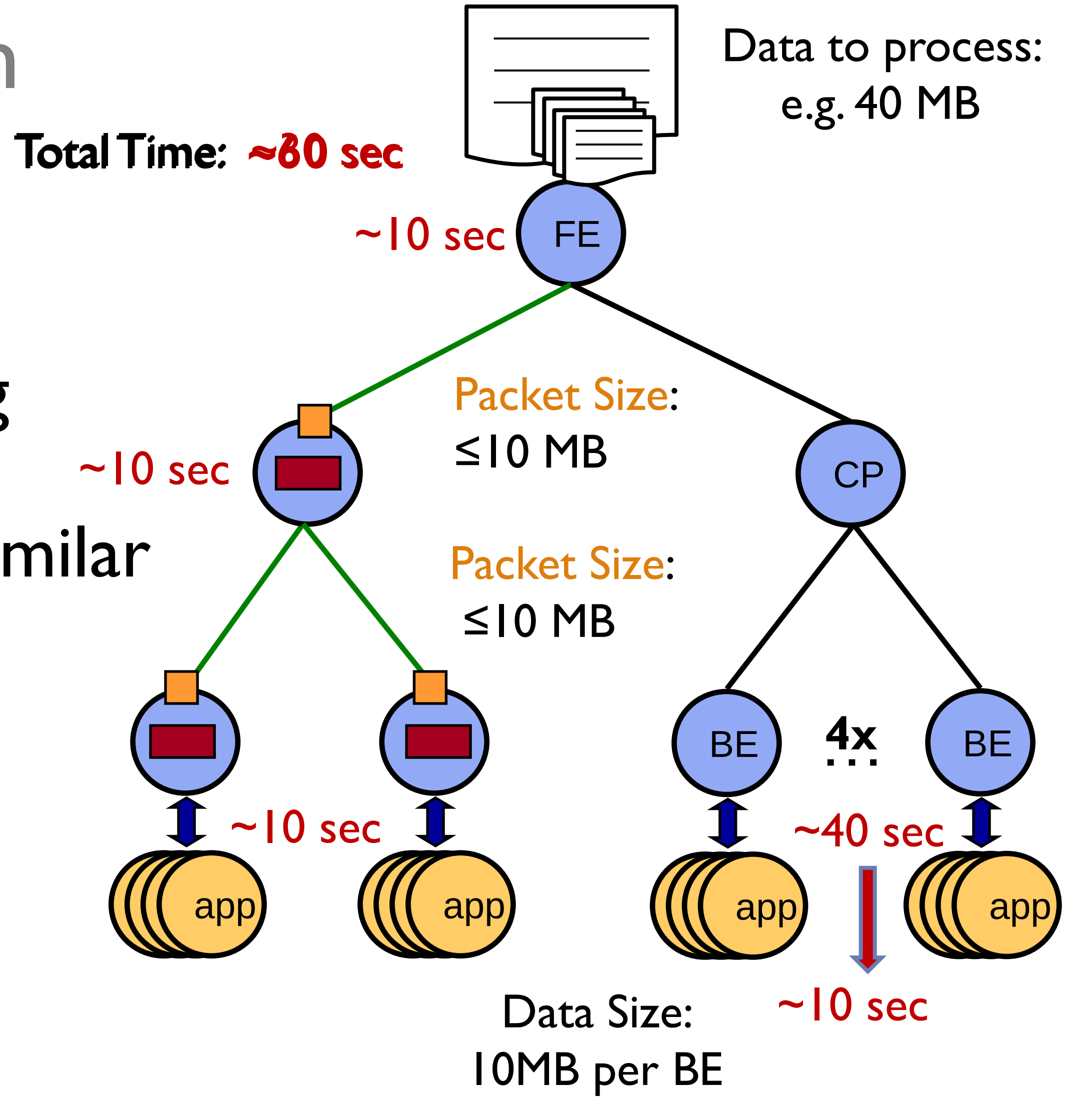
- General-purpose TBON API
  - Network: user-defined topology
  - Stream: logical data channel
    - to a set of back-ends
    - multicast, gather, and custom reduction
  - Packet: collection of data
  - Filter: stream data operator
    - synchronization
    - transformation
- Widely adopted by HPC tools
  - CEPBA toolkit
  - Cray ATP & CCDB
  - Open|SpeedShop & CBTF
  - STAT
  - TAU



TBON Computation

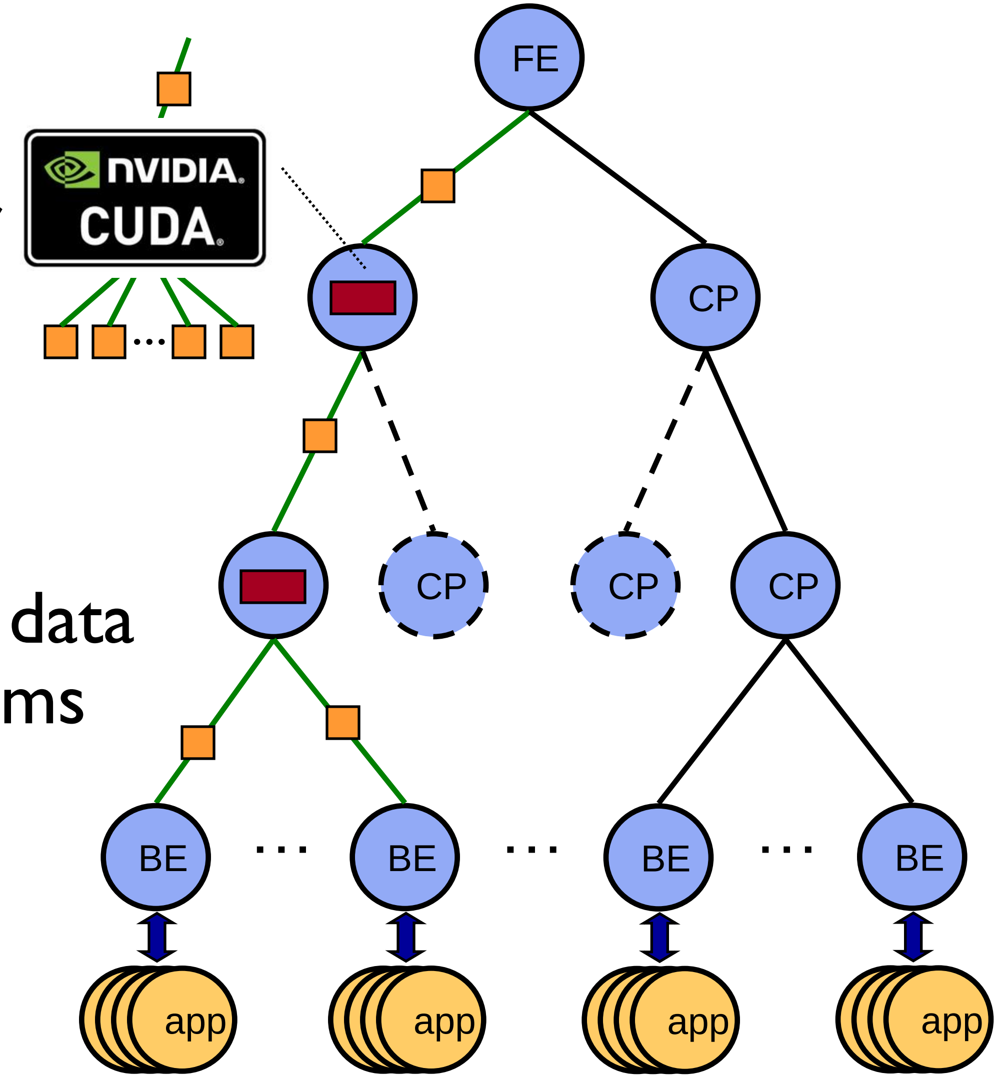
Ideal Characteristics:

- Filter output size constant or decreasing
- Computation rate similar across levels
- Adjustable for load balance



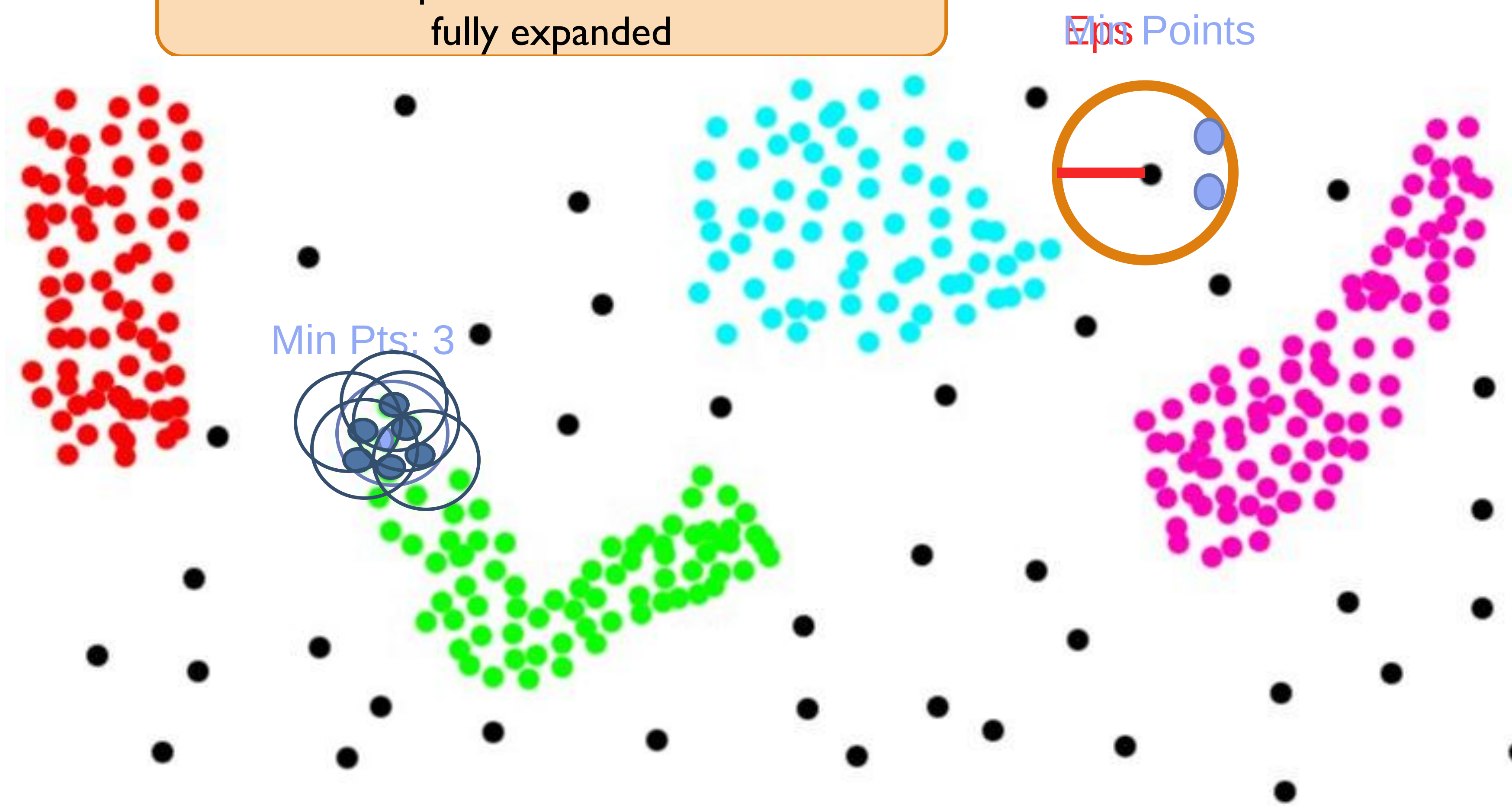
Why GPUs?

- Natural fit
- Increase compute power
- Trade computation for bandwidth
  - Derived summaries
  - Compute and send Δ data
  - Compression algorithms (e.g. LZO, zlib, etc.)



Clustering Example (DBSCAN[1])

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



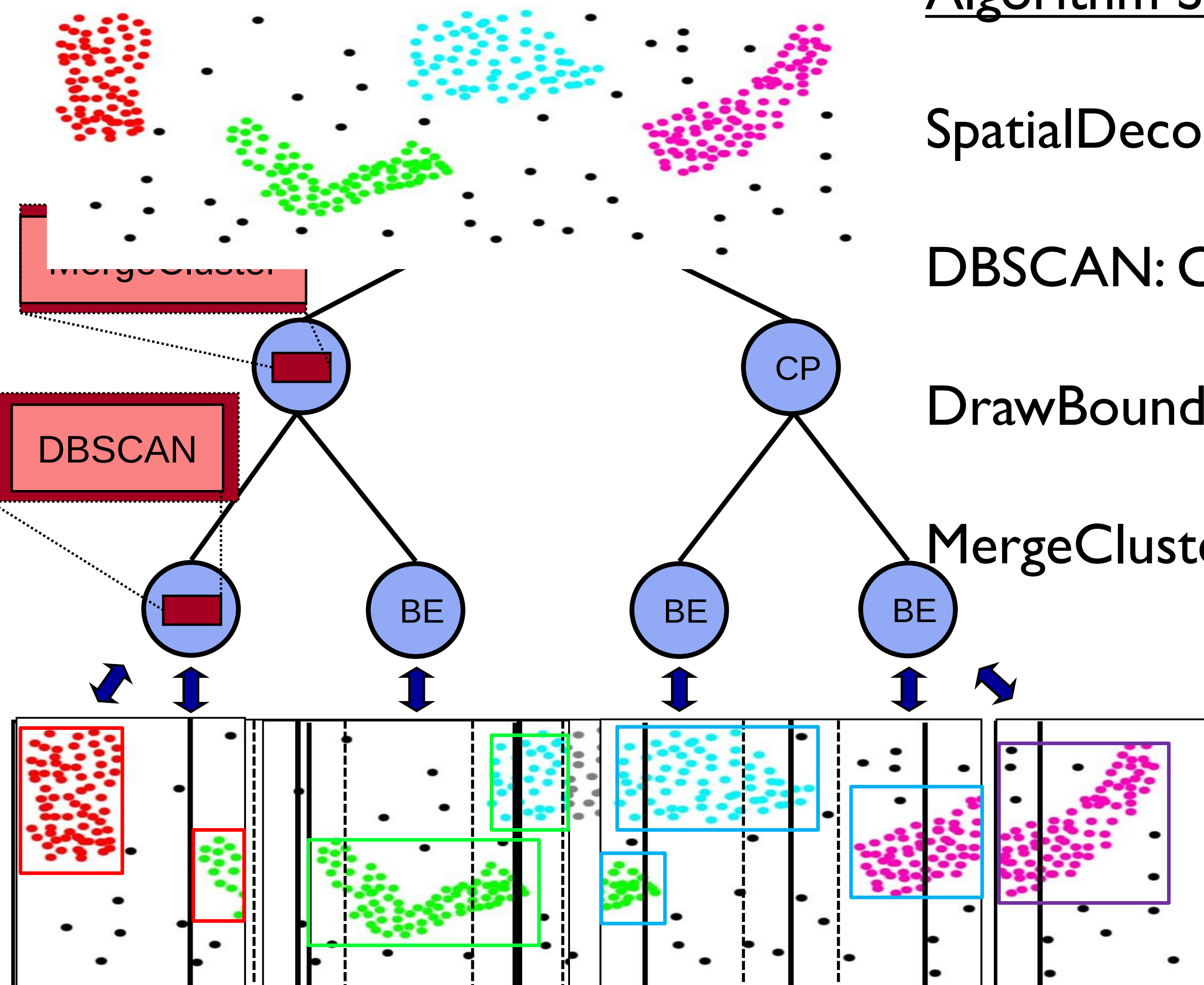
[1] M. Ester et. al., A density-based algorithm for discovering clusters in large spatial databases with noise, (1996)

Scaling DBSCAN

- PDBSCAN[2]
  - Quality equivalent to single DBSCAN
  - Linear speedup up to 8 nodes
- DBDC[3]
  - Sacrifices quality
  - ~30x speedup on 15 nodes
- CUDA-Dclust[4]
  - Quality equivalent to DBSCAN
  - ~15x faster on 1 node

[2] X. Xu et. al., A fast Parallel Clustering Algorithm for Large Spatial Databases (1999)
[3] E. Januzaj et. al., DBDC: Density Based Distributed Clustering (2004)
[4] C. Bohm et al., Density-based clustering using graphics processors (2009)

Tree-Based Clustering: Mr. Scan



<u>Algorithm Steps</u>

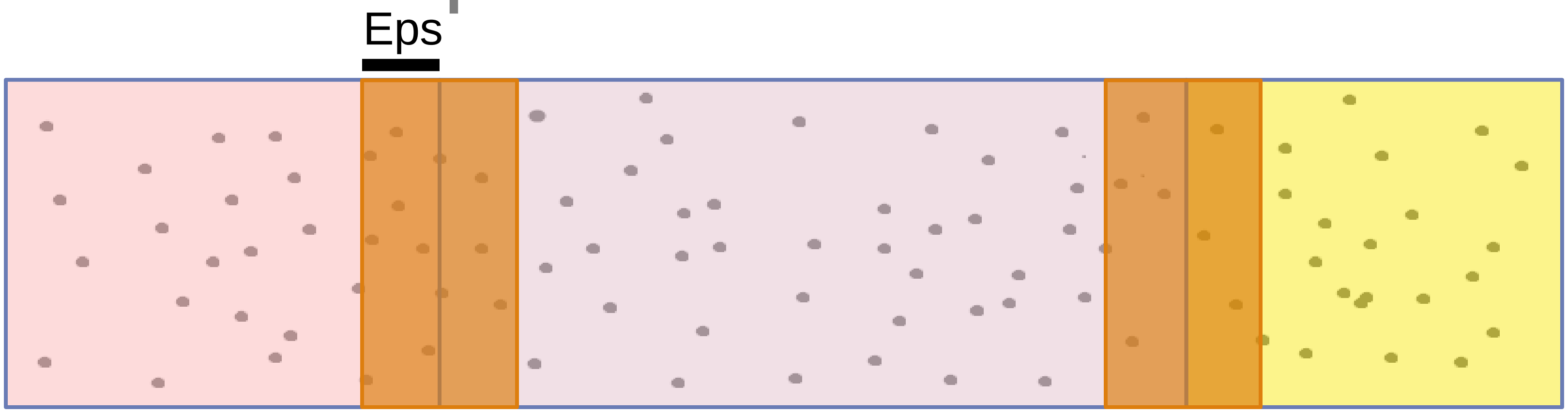
SpatialDecomp: CPU(@ FE)

DBSCAN: CPU or GPU(@ BE)

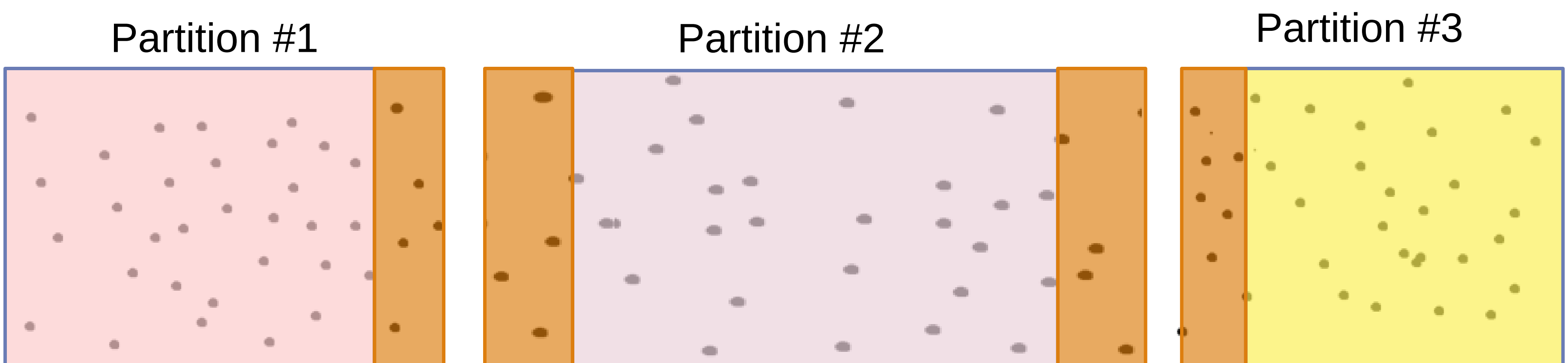
DrawBoundBox: CPU or GPU

MergeCluster: CPU (x #levels)

Spatial Decomposition

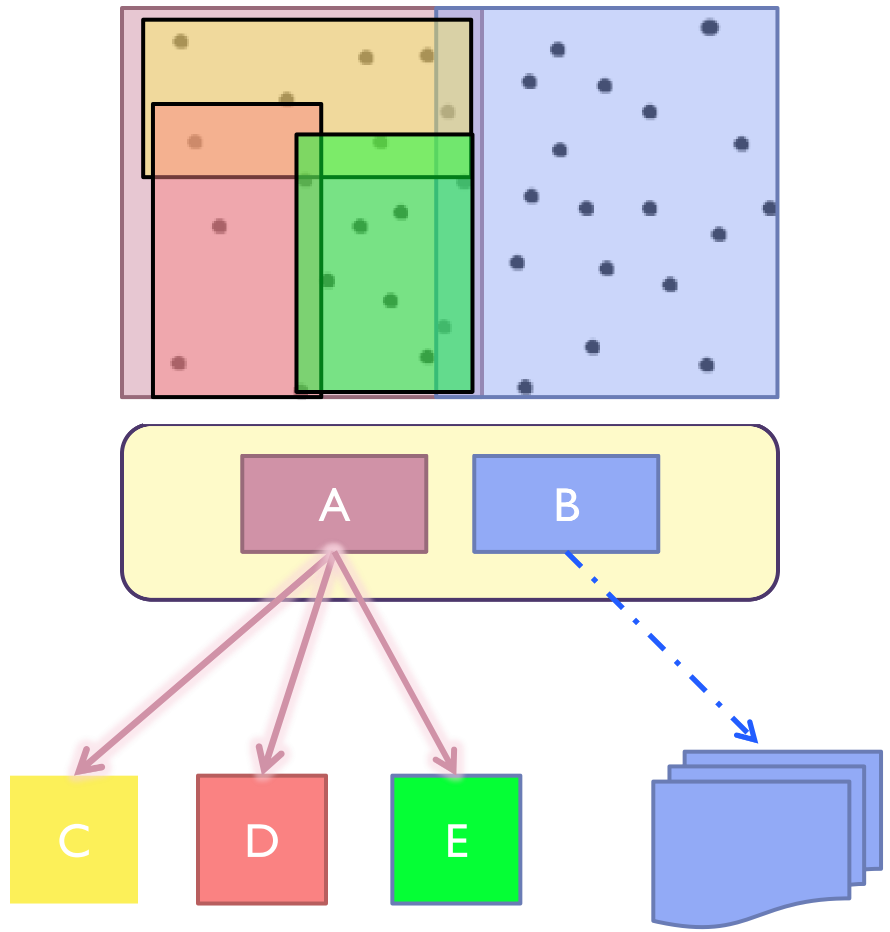


1. Start with an input of Spatially Referenced points
2. Partition the region into equal sized density regions across one dimension
3. Add the shadow region area of one Epsilon to all density regions



DBSCAN - CPU

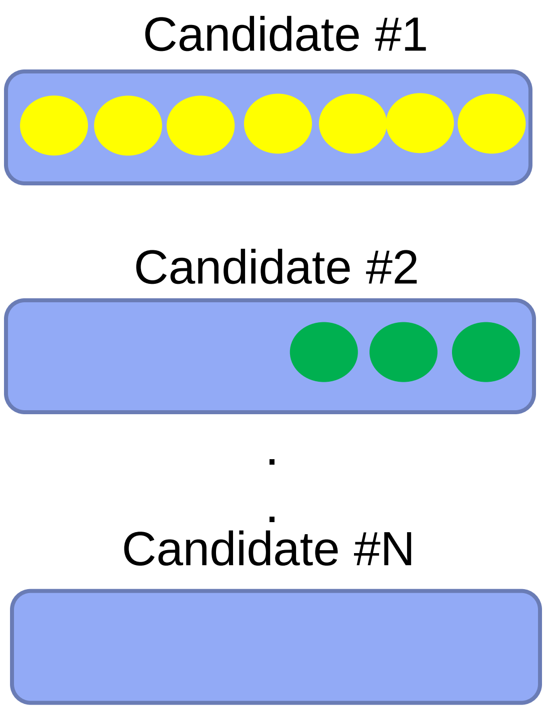
- Run on local slice for each BE
- R* tree
- Start at random point
- Cluster w/ respect to Eps, MinPts
- Complexity: O(n log n)



R* Tree Example

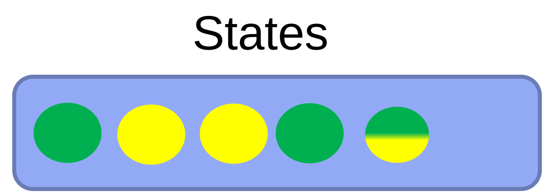
GPU DBSCAN Filter

Candidate Clusters are potential clusters being explored concurrently in the GPU

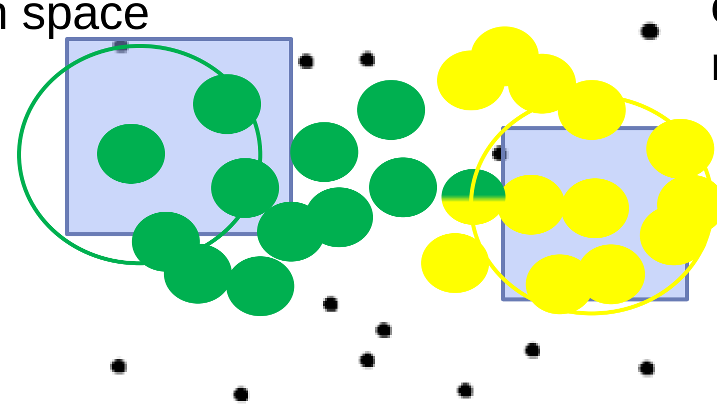


Number of cluster candidates is limited by GPU Characteristics

State array stores the current state of points in the search space



Coarse grain search space

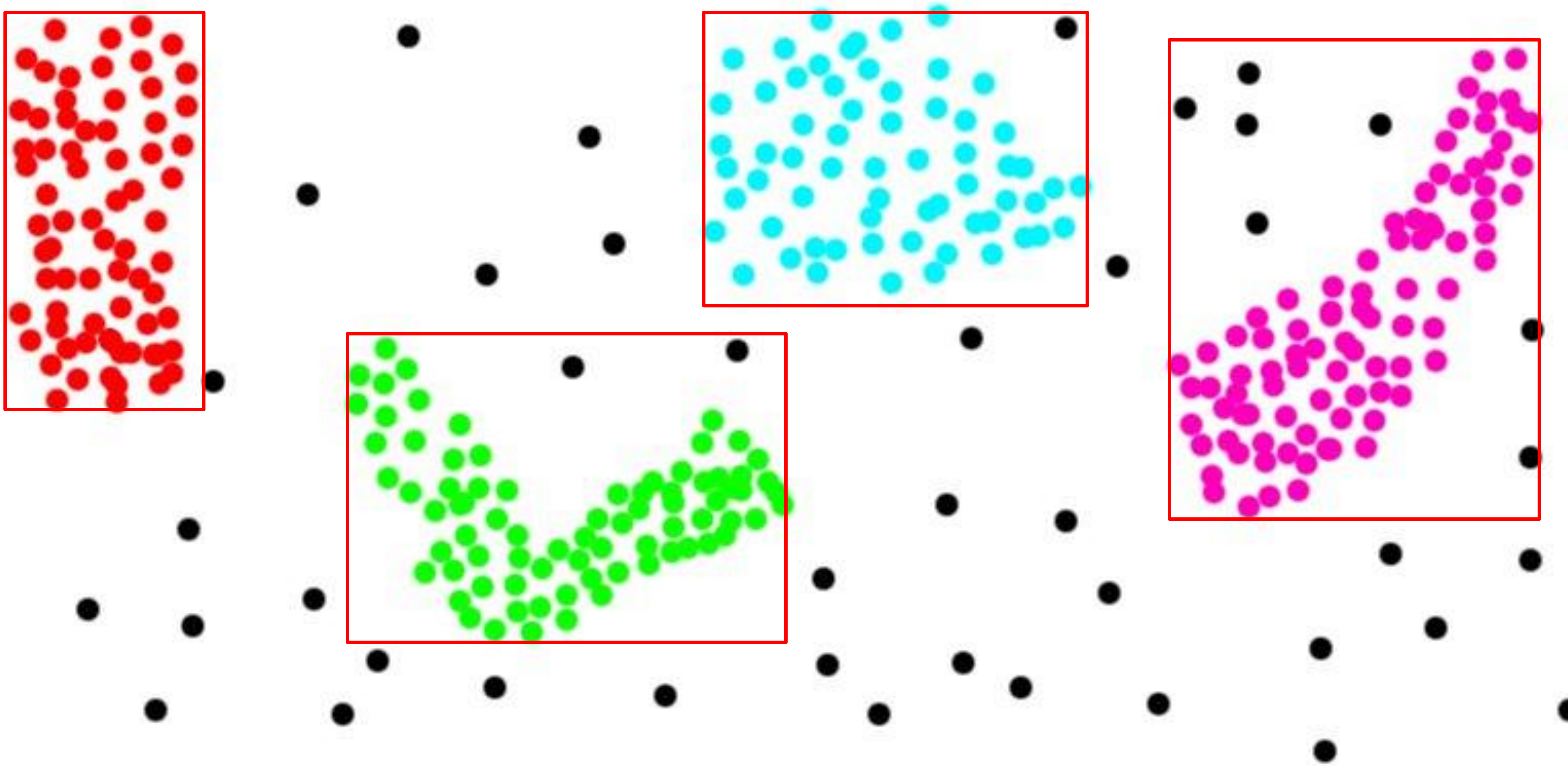


Chain Collisions causing cluster merges

GPU DBSCAN operates similarly to the CPU version with two exceptions

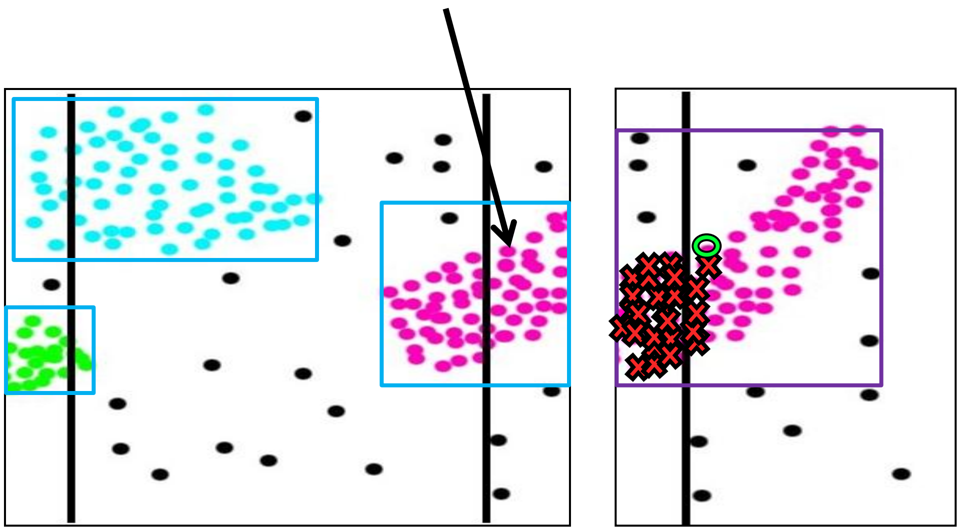
CUDA-DCLUST [09 – Böhm]

DrawBoundBox – CPU || GPU



MergeBoundBox - CPU

- Checks for merge if box within shadow
- At least one point MUST be in common
- Iterate through ALL points in right cluster

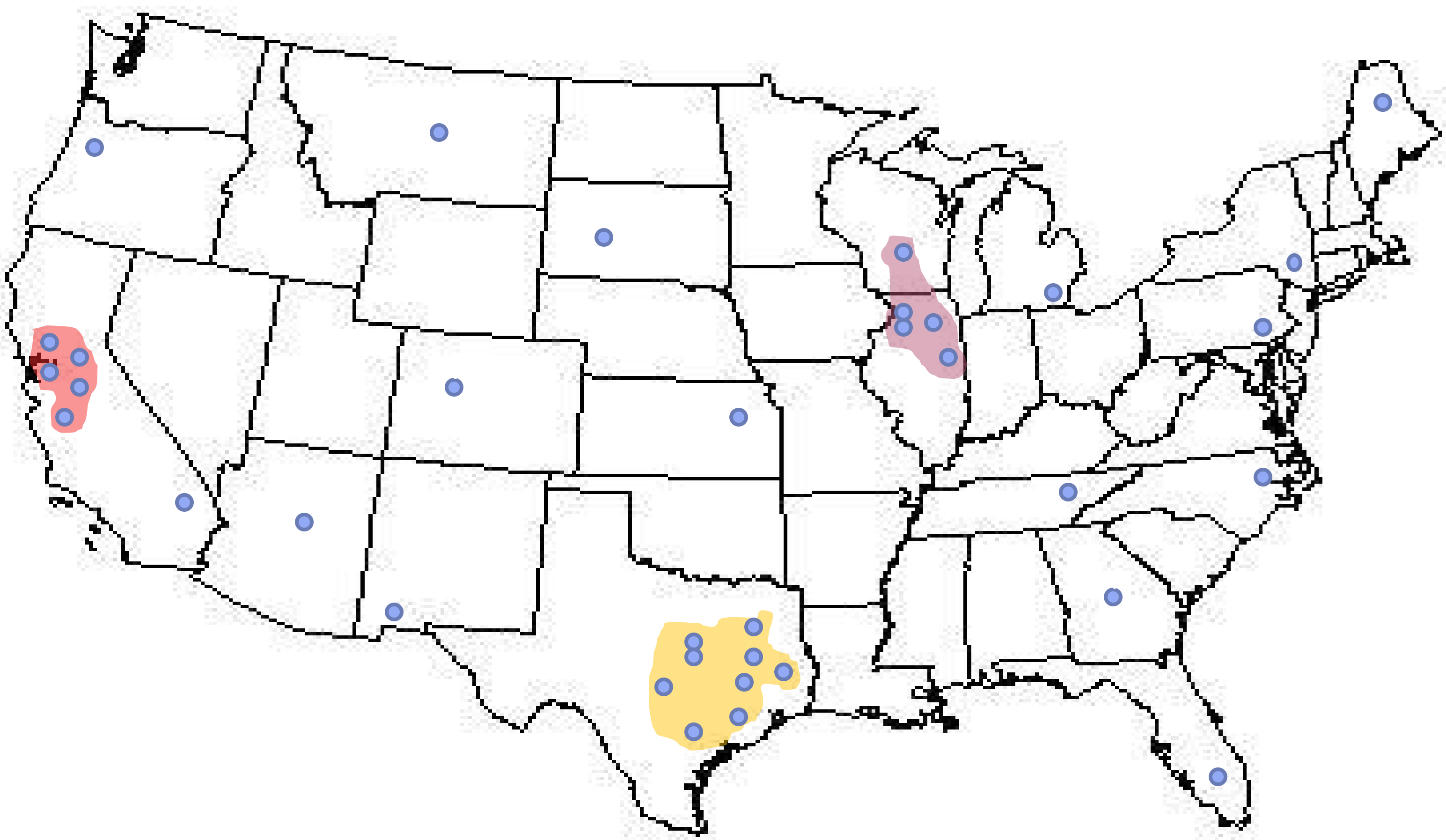


Match!

The Tweet Stream

Kayla Lorelle @kbombbbb 5h

Why did I have to get the **flu** :(

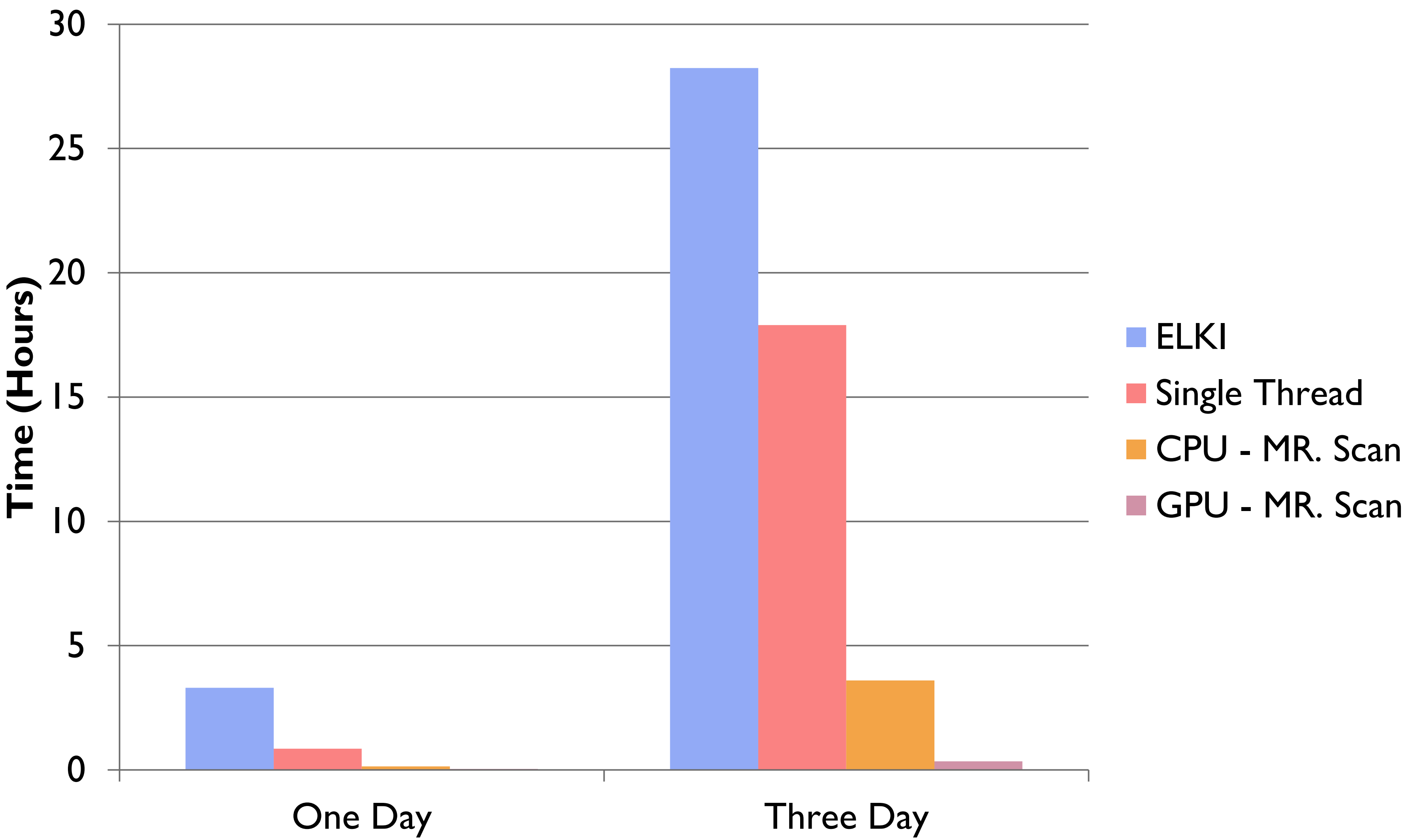


Source: Twitter, Map: About.com

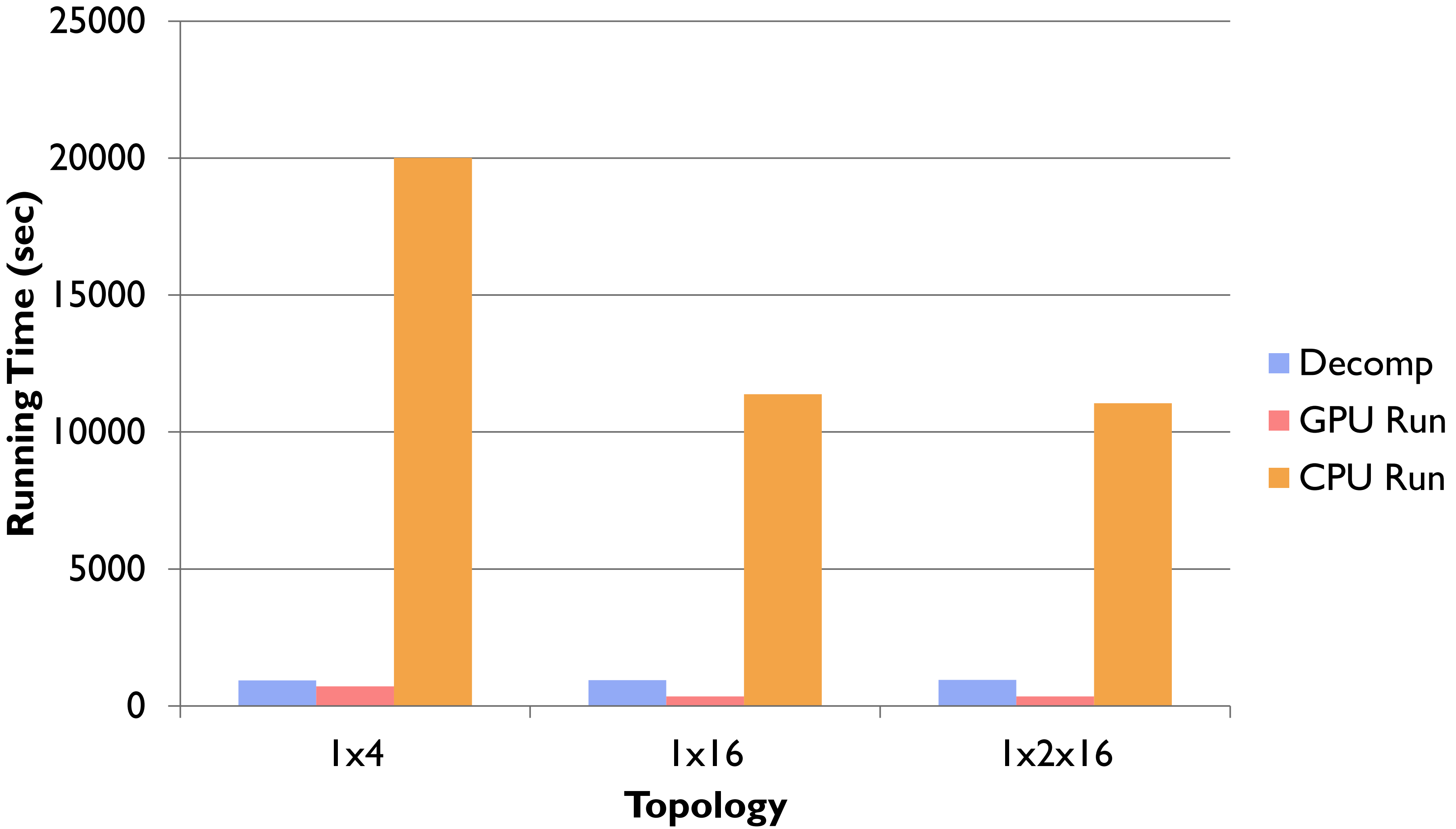
Evaluation

- Dataset: 1-3 "Tweet Days"
- Measuring:
  - Time to completion
  - Quality compared to single-threaded DBSCAN
- Algorithms:
  - Single-Threaded DBSCAN
  - DBDC
  - MRNet w/DBSCAN filter
  - MRNet w/DBSCAN GPU filter

Results

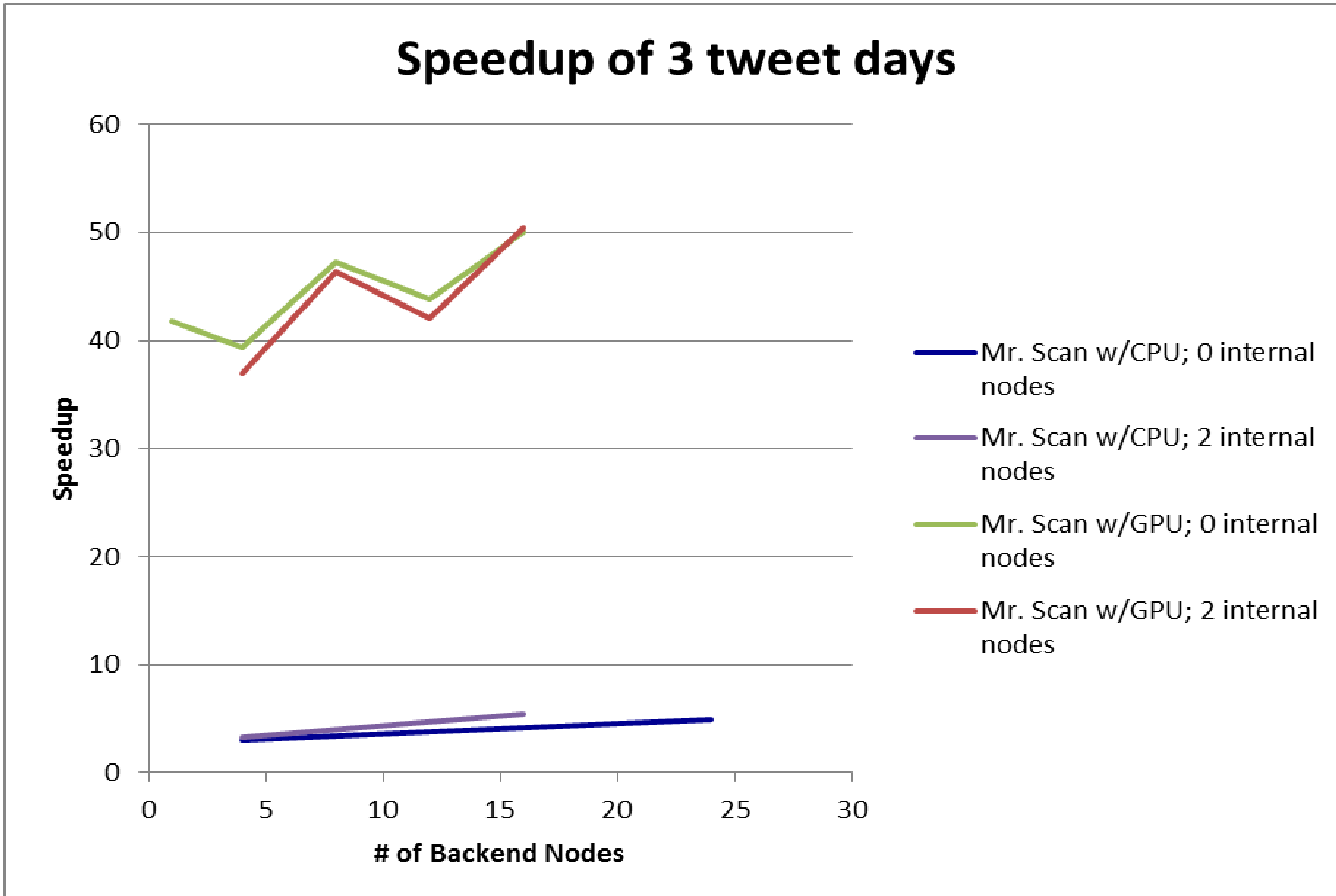

30
25
20
15
10
5
0
Time (Hours)
One Day
Three Day
ELKI
Single Thread
CPU - MR. Scan
GPU - MR. Scan

Results



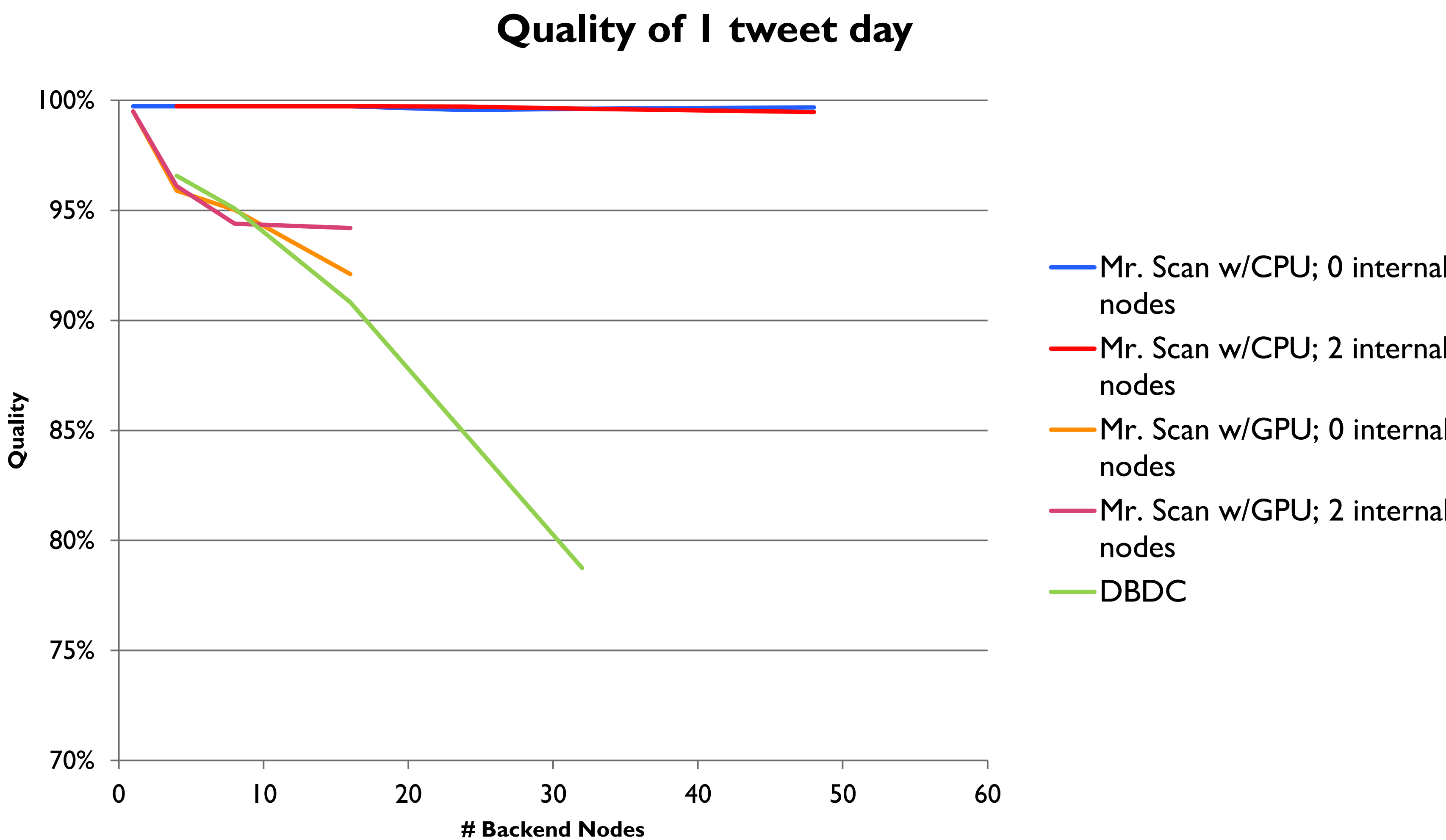
25000
20000
15000
10000
5000
0
Running Time (sec)
1x4
1x16
1x2x16
Topology
Decomp
GPU Run
CPU Run

Results



Speedup of 3 tweet days
Speedup
60
50
40
30
20
10
0
0
5
10
15
20
25
30
# of Backend Nodes
Mr. Scan w/CPU; 0 internal nodes
Mr. Scan w/CPU; 2 internal nodes
Mr. Scan w/GPU; 0 internal nodes
Mr. Scan w/GPU; 2 internal nodes

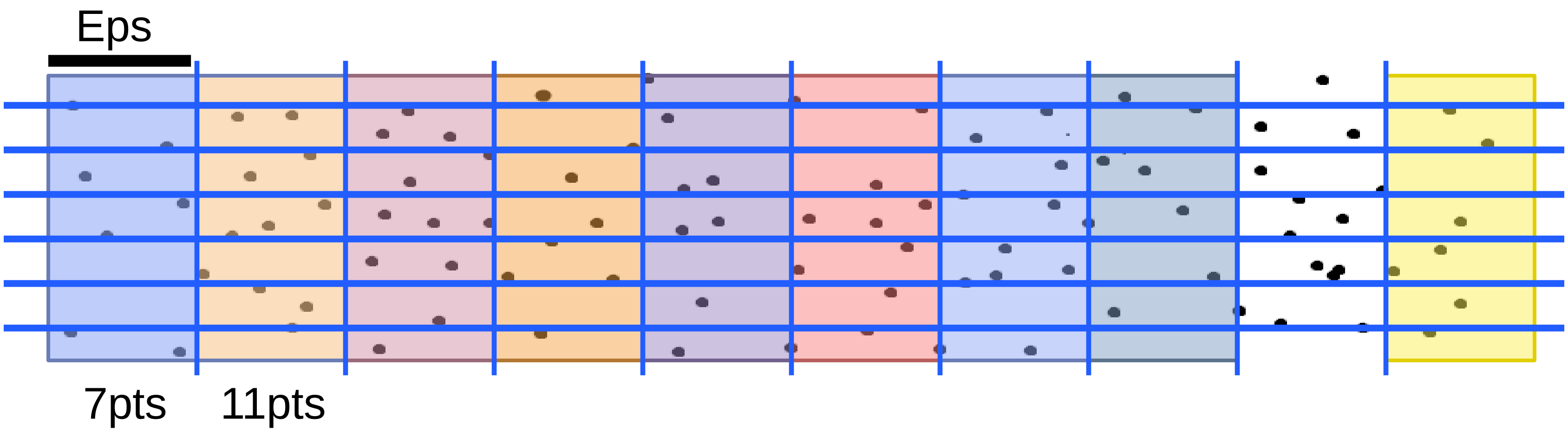
Quality


Quality of 1 tweet day
100%
95%
90%
85%
80%
75%
70%
Quality
0
10
20
30
40
50
60
Backend Nodes
Mr. Scan w/CPU; 0 internal nodes
Mr. Scan w/CPU; 2 internal nodes
Mr. Scan w/GPU; 0 internal nodes
Mr. Scan w/GPU; 2 internal nodes
DBDC

Future Work

- Scaling Issues
  - Spatial Decomposition
  - Merging Algorithm

2D Spatial Decomposition

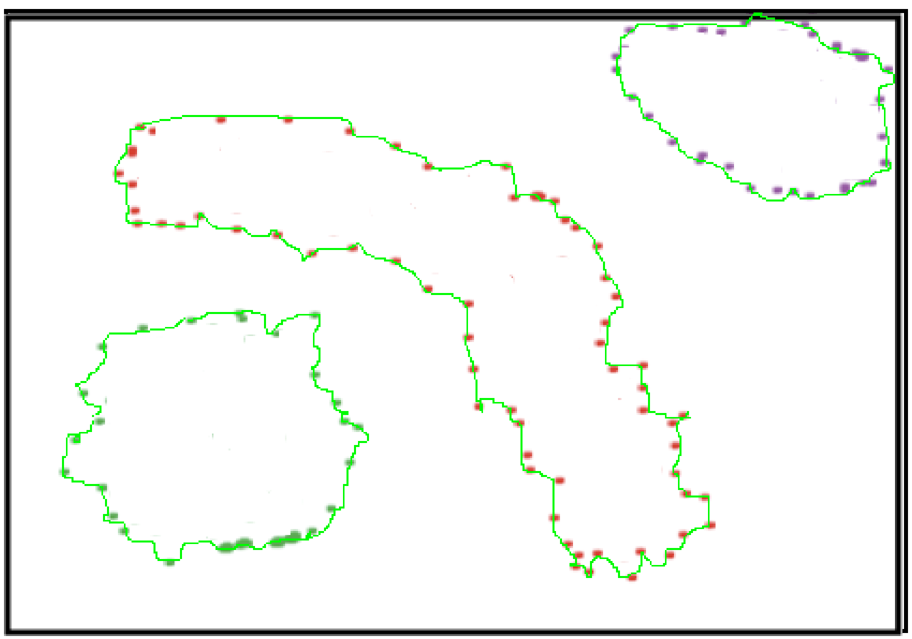


- 1D Spatial Decomposition has some severe limitations
  - Splits can have wildly differing point counts
  - Number of splits limited by Epsilon

- 2D Spatial Decomposition would allow for more Splits with more equal point counts

Merging Algorithm

- Bounding Box
  - No quality degradation
  - Limits iteration, but still iterates



- Possible Alternative: Concave Hull
  - DBSCAN on border points
  - Avoids iteration
  - Constant output size
  - $O(n^2)$ on BEs

Use Cases

- Twitter Data
  - "Flu Tweets"
  - Mood/Topic clustering
  - Riot prediction
- Any Spatial data
  - Currently limited to 2D

Conclusion

- DBSCAN performance scales
- Quality able to be maintained
- Goal: Scale O(100,000 nodes)