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

---

Tree-Based Density Clustering using Graphics Processors
TBON Computation

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

Data to process: e.g. 40 MB

Packet Size: ≤10 MB

Data Size:
- e.g. 40 MB

Total Time:
- ~80 sec
- ~10 sec

Packet Size:
- ≤10 MB

Packet Size:
- ≤10 MB

Data Size:
- 10MB per BE

~40 sec

~10 sec

~10 sec

~10 sec

~4x
Why GPUs?

- Natural fit
- Increase compute power
- Trade computation for bandwidth
  - Derived summaries
  - Compute and send $\Delta$ data
  - Compression algorithms (e.g. LZO, zlib, etc.)
Goal: Find regions that meet minimum density and spatial distance characteristics.

The two parameters that determine if a point is in a cluster is $\epsilon (\text{Eps})$ and $\text{MinPts}$. If the number of points in $\epsilon$ is $\text{MinPts}$, the point is a core point.

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

Clustering Example (DBSCAN$^{[1]}$)

Min Pts: 3

Scaling DBSCAN

- **PDBSCAN**\(^2\)
  - Quality equivalent to single DBSCAN
  - Linear speedup up to 8 nodes

- **DBDC**\(^3\)
  - Sacrifices quality
  - \(~30\times\) speedup on 15 nodes

- **CUDA-Dclust**\(^4\)
  - Quality equivalent to DBSCAN
  - \(~15\times\) 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

Algorithm Steps

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

Candidate #1

Candidate #2

Candidate #N

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

States

Coarse grain search space

Chain Collisions causing cluster merges

GPU DBSCAN operates similarly to the CPU version with two exceptions

Number of cluster candidates is limited by GPU Characteristics

CUDA-DCLUS [09 – Böhm]
Tree-Based Density Clustering using Graphics Processors
MergeBoundBox - CPU

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

Kayla Lorelle @kbombbb
Why did I have to get the flu :(  

Map: About.com

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

One Day

Time (Hours)

Three Day

- ELKI
- Single Thread
- CPU - MR. Scan
- GPU - MR. Scan

Tree-Based Density Clustering using Graphics Processors
Results

Running Time (sec)

- Decomp
- GPU Run
- CPU Run

Topology

1x4
1x16
1x2x16
Results

Speedup of 3 tweet days

Speedup

# 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 of 1 tweet day

Quality

# 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 $O(100,000 \text{ nodes})$