JÜLICH
FORSCHUNGSZENTRUM

MLHPC 2015

HPDBSCAN – Highly Parallel DBSCAN

Markus Götz, Christian Bodenstein and Morris Riedel
Jülich Supercomputing Center (JSC) // University of Iceland

Outline

Introduction

- DBSCAN
- Related Work

Highly Parallel DBSCAN

- Parallelization Strategy
- Implementation
- Performance Evaluation

Conclusion

- Use Case – Point Clouds
- Discussion

DBSCAN

Density **b**ased **s**patial **c**lustering for **a**pplication with **n**oise

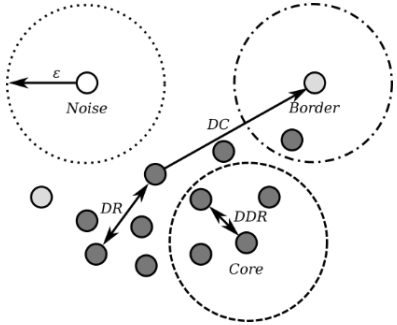
- Formulated 1996 by Ester et. al. (TUM, Germany)
- Unsupervised clustering algorithm

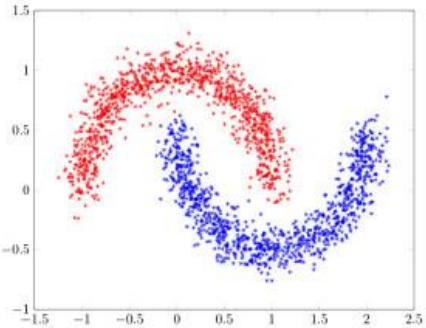
Parameters

- *epsilon (ε)* – spatial search radius
- *minPoints* – density threashold

Properties

- Maximize local point density recursively
- Detects arbitrarily shaped clusters (except „bow-ties“)
- Filters signal for noise





Related Work

Earlier attempts

- Around 1999-2000 (three years after initial proposition)
- Focused mainly on parallelizing neighborhood queries
  - Distributed Indices
  - Works well for shared-memory/small number of core systems
- Dozens of papers/approaches

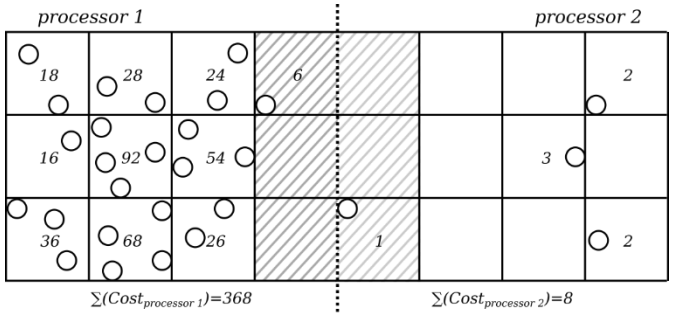
Recents attempts

- 10 years break – no papers
- Resurge of interest with the Big Data hype
- Stronger focus on scalability, most notable work from Patwary et. al., SC2013

Parallelization Strategy

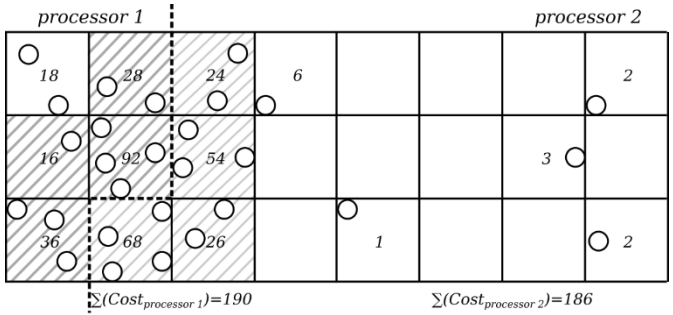
Data Decomposition

- Essentially spatial problem
- Constant search distance
- Overlay data space with cells and split
- Requires data redistribution



Load balancing

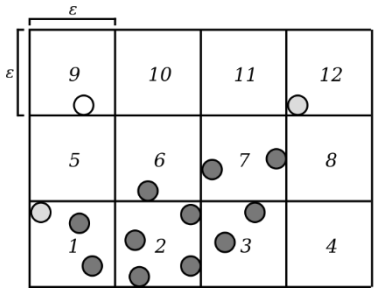
- Estimate computational cost per cell (complexity)
- Product of neighborhood size and cell size
- Sum up grand total over all processors
- Assign cells p^{th} part of sum per processor



Parallelization Strategy

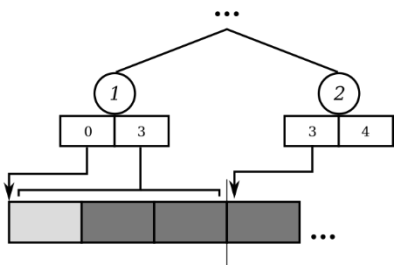
Local DBSCAN

- Data is sorted after redistribution
- Run-length index (constant lookup time)
- Execute standard DBSCAN on local chunk



Merging

- Exchange halo regions
- Find differences in cluster labeling of core points
- Generate label mapping rules
- Exchange rules globally and apply
- Restore initial data order



Implementation

Source code

- C++ MPI+OpenMP hybrid application
- Facilitates HDF5 for data I/O
- Can be used as CL application or shared-library
- https://bitbucket.org/markus.goetz/hpdbscan

Lock-free cluster label assignment

- Cluster labels are stored consecutive in memory
- Need for synchronization between threads
- Use `atomicMin` on encoded labels (`CLLLL…LL`)

Performance Evaluation

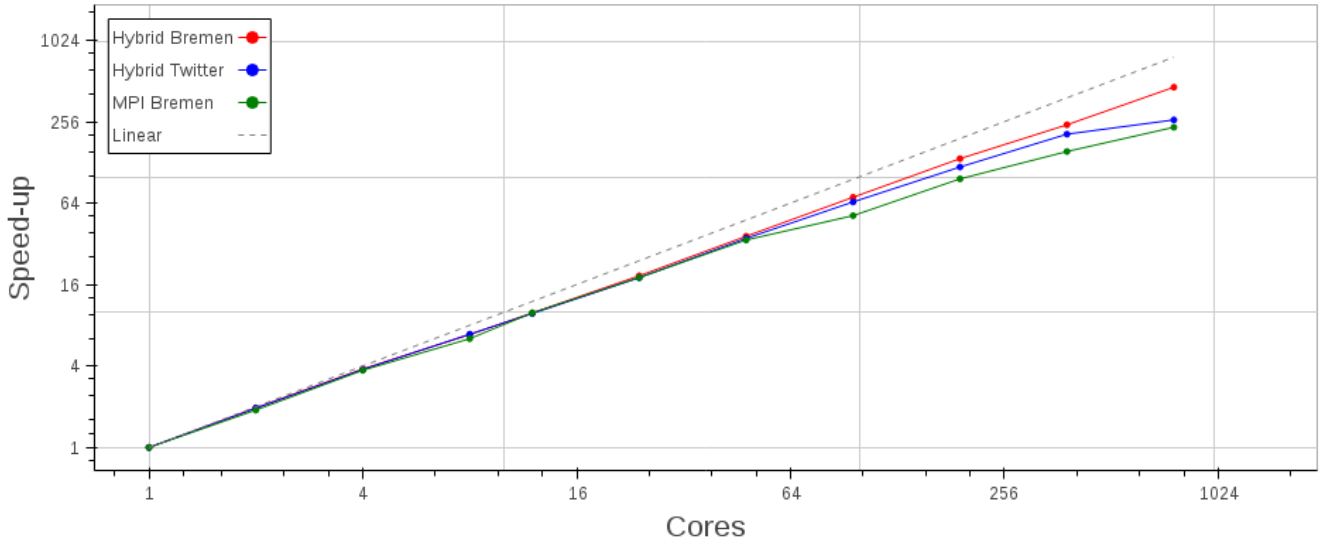
Datasets

- Bremen point cloud (~82 million entries)
- Collection of tweets (~17 million entries)

Environment

- JuDGE supercomputer at JSC
  - Intel Xeon 12-core CPUs
  - DDR3 memory
  - Infiniband
- Scaled up to 768 cores (1/4th)
- Dedicated resources

Performance Evaluation

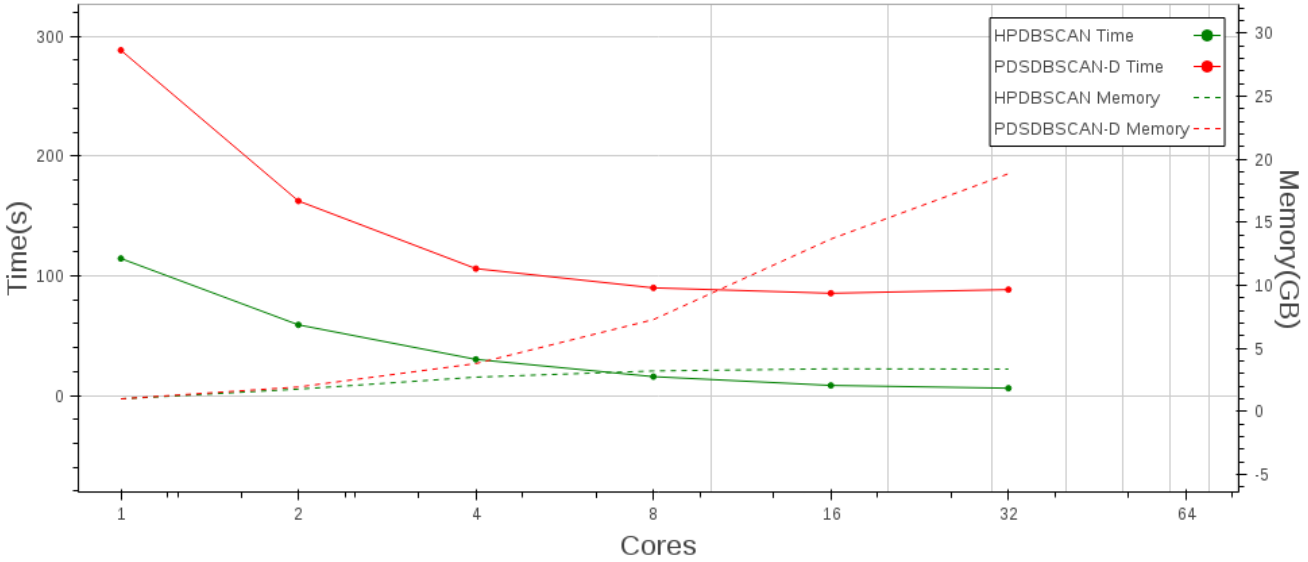


Performance Evaluation

Comparison with PDSDBSCAN-D

- Presented at SC2013
- Parallel DBSCAN based on disjoint-sets
  - MPI version evaluated
  - Binary mode
- Only other with published source (thank you!)
- Results
  - Better computation time
  - Orders of magnitude better memory usage

Performance Evaluation



Use Cases – Point Clouds

Point clouds

- 3D to 5D laser reflection scans
- Captured by robots or drones
- Order of million to billion entries

Application

- Identify (segment) structures
- Resilient to noise (e.g. animals)
- Survey Roman ruins on Cyprus
  - Identify dig sites
  - Build model of known objects

Contact: ***m.goetz@fz-juelich.de***

Slides: Send me a mail with a request