Parallel DBSCAN

Team members: Bowen Chen

Url

https://vertexc.github.io/parallel-dbscan/

Summary

In this project, we are going to implement an efficient clustring method called DBSCAN in parallel.

Background

There are multiple clustring methods, like K-Means, Hierarchical Clustering, and many others. Among them, DBSCAN is a density-based clustering algorithm with some notable advantages. It doesn't requires predifined number of clusters like K-Means. Meanwhile, it identifies outliers as noises and works well on arbitrarily sized and shaped clusters.

Here is how it works sequentially

```
// pseudocode from https://en.wikipedia.org/wiki/DBSCAN
DBSCAN(DB, distFunc, eps, minPts) {
    C := 0                                                    /* Cluster counter */
    for each point P in database DB {
        if label(P) ≠ undefined then continue                 /* Previously processed in inner loop */
        Neighbors N := RangeQuery(DB, distFunc, P, eps)       /* Find neighbors */
        if |N| < minPts then {                                /* Density check */
            label(P) := Noise                                 /* Label as Noise */
            continue
        }
        C := C + 1                                            /* next cluster label */
        label(P) := C                                         /* Label initial point */
        SeedSet S := N \ {P}                                  /* Neighbors to expand */
        for each point Q in S {                               /* Process every seed point Q */
            if label(Q) = Noise then label(Q) := C            /* Change Noise to border point */
            if label(Q) ≠ undefined then continue             /* Previously processed (e.g., border point) */
            label(Q) := C                                     /* Label
neighbor */
            Neighbors N := RangeQuery(DB, distFunc, Q, eps) /* Find
neighbors */
            if |N| ≥ minPts then {                          /* Density
check (if Q is a core point) */
                S := S ∪ N                                  /* Add new
neighbors to seed set */
            }
        }
    }
}

RangeQuery(DB, distFunc, Q, eps) {
    Neighbors N := empty list
    for each point P in database DB {                      /* Scan all
points in the database */
        if distFunc(Q, P) ≤ eps then {                     /* Compute
distance and check epsilon */
            N := N ∪ {P}                                   /* Add to
result */
        }
    }
    return N
}
```

Basically, it starts with an randomly selected starting point, all points within distance of **epsilon ε** is considered as **neighborhood** . If number of points in neighborhood is larger than **minPts**, we start to mark this point with a new cluster label and further explore its neighborhood, otherwise, we mark the point as **noise**. We iterate the process until all points is properly labeled.

Goals and Deliverables

Plan to Achive

We plan to at least achive speed up compared to the baseline (sequential DBSCAN).

Hope to Achive

We hope we can further looked into more advanced implementations of DBSCAN, and try to further optimize it. (Either propose some new parallel algorithm, or fine-tune on the given platform)

Platform Choice

The project is going to be implemented in C++, Cuda/MPI.

We are going to evalaute the performance on our own desktop. (CPU:Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz, GPU: NVIDIA Corporation GP106 [GeForce GTX 1060 3GB])

The Challenges

First of all, it is non-trivial to write efficient parallel code for DBSCAN. We can notice that **RangeQuery** is the computationally intensive part where we can potentially make it parallel. However, for the higher level of parallism, we have to fix the starting point and search on top of that, having multiple starting points may cause the incorrect clustering results. Meanwhile, when search across the numbers, we can notice that the algorithm is growing the neighbors dynamically, where the load balancing and synchronization could be an issue.

Resources

Here are some search paper on DBSCAN we may look into

- A new scalable parallel DBSCAN algorithm using the disjoint-set data structure (SC 2012)
- G-DBSCAN: A GPU Accelerated Algorithm for Density-based Clustering (ICCS 2013)
- HPDBSCAN: highly parallel DBSCAN (SC 2015)
- Theoretically-Efficient and Practical Parallel DBSCAN (SIGMOD 2020)

Schedule

The project's implementation basically contains two parts, one is python level multi-worker data process, and another is cuda kernels implementation.

- **Week2 (11.15-11.21)**, we will implement a sequential verison of DBSCAN in C++ as baseline and correctness validation. In the meantime, we will explore different ways to implement DBSCAN in parallel.
- **Week3 (11.22-11.28)**, we will implement parallel DBSCAN and compare it to the baseline.
- **Week4 (12.29-12.5)**, we will explore more advanced DBSCAN algorithm from research papers and see if we can further improve the performance.
- **Week5 (12.6-12.10)**, we will wrap up the code, finish the final report and prepare for the poster session.