Project Proposal

15–618 (Fall '16)

Submitted: Monday, October 31st

Anant Subramanian
assubram@andrew.cmu.edu

Feroze Naina
fdheenmo@andrew.cmu.edu

Spectral Graph Clustering using MPI + CUDA

1 Summary

We intend to implement a hybrid MPI + CUDA algorithm to perform spectral clustering of an input graph. We will test the algorithm against a sequential MATLAB implementation and a parallel single-GPU CUDA implementation using one (or more) of the SNAP data sets [1] on two fronts: quality and overall computation time. We may use more precise measurements to support our claims and verify our hypotheses.

2 Background

Given an undirected similarity graph $G = (V, E, W)$, a clustering of the graph is an assignment of vertices v_i, $\forall i \in \{1, 2, \cdots, |V|\}$ to clusters c_i, $\forall i \in \{1, 2, \cdots, C\}$ such that a particular *cost metric* on the similarities $W(v_i, v_j)$ over the clusters c_i is minimized.

A *normalized cut* of a graph is a partition of the graph into two partitions A and B such that the sum of the weights of the edges connecting vertices in A to those in B is the minimum of all possible partitions *and* the size of A and B are similar. Mathematically,

$$\text{Normalized Cut}\,(A, B) = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{vol(A)} + \frac{1}{vol(B)} \right)$$

where

$$vol(A) = \sum_{i \in A} d_i$$

Since this problem is NP-hard in nature, the approach of spectral clustering attempts to relax the constraints of the normalized cut and approximate the normalized cut using graph Laplacians.

Without diving into the mathematical details (see [2, 3, 4] for more details), the problem reduces to that of constructing the graph Laplacian (or the normalized graph Laplacian):

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \qquad (\mathbf{D} = \text{Diagonal degree matrix}, \mathbf{A} = \text{Affinity matrix})$$
$$\mathbf{L}^{norm} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$$

and computing the two partitions using the second-smallest eigen value corresponding to one of the two Laplacian matrices. One popular method for computing the eigen values is using the Lanczos [5] algorithm, which involves iterative matrix-vector and vector-vector computations.

Therefore, from an engineering standpoint, spectral graph clustering algorithms reduce to steps of matrix operations.

Since GPUs are excellent candidates for speeding up matrix operations by performing independent computations in parallel, we intend to use them to speed up steps of the spectral clustering algorithm as well. Further, we intend to discover methods to distribute and perform these computations on a cluster of nodes using MPI, while continuing to exploit the parallelism offered by the GPUs on each of these nodes. The following figure illustrates our parallelization strategy.

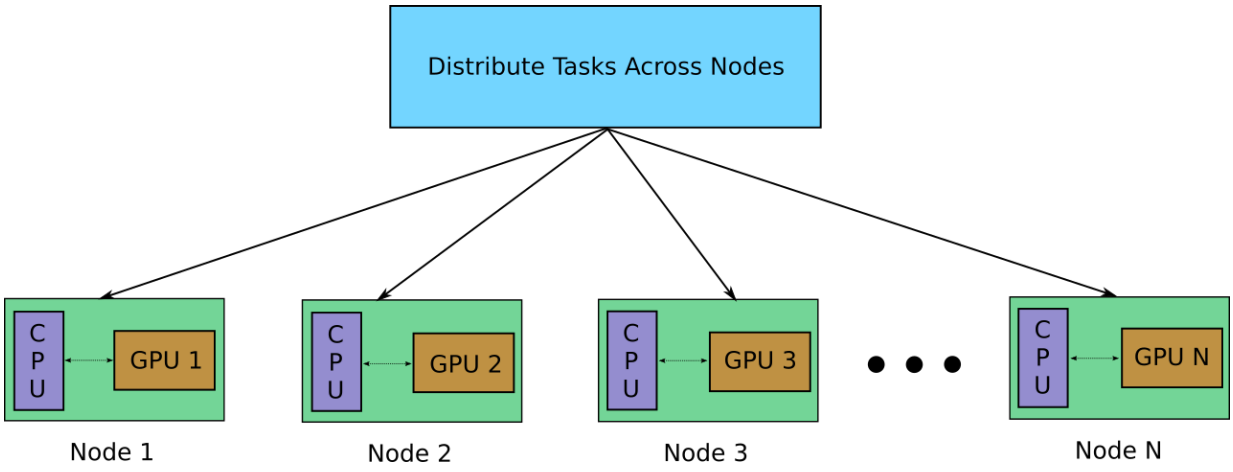

Figure 1: The hybrid parallelization strategy that will be used by our algorithm.

To the best of our knowledge, this is the first such effort in this direction.

3 Challenges

- Spectral clustering is **mathematically complex**. It would take us time to understand the mathematics well enough to argue about the correctness of our results. This is not just from a purely mathematical perspective, but also from a systems point of view, as we are also limited by the numerical precision that our systems have to offer.

- Since we are using a hybrid parallelization approach, there will potentially be a **lot of communication overhead**. It is not easy to see why this might work until we have explored the best possible ways to reduce the communication overhead. We could potentially compute and store data that doesn't change once at the beginning of the algorithm, but we might then be limited by the available memory.

- For a GPU on a single cluster node, we can view the **data as being stored in a NUMA system**:
  Device Memory → Main Memory → Other Node's Main Memory → Other Node's Device Memory.
  We might have to explore concepts such as pre-fetching/latency hiding/pipelining to improve performance.

- There are a **lot of possible parallel approaches that we could explore**. We could attempt to perform different steps of the algorithm in parallel, or we could operate in parallel on different parts of the data for the same algorithm step. For matrix operations, we could explore row-distributed representations, column-distributed representations, 2D-distributed representations, etc. We would have to narrow down on the best working approach fast enough to have enough time to refine it.

- Since we are **attempting to get two different frameworks to work together**, we might run into unforeseen issues in getting MPI and CUDA to work well together. We would have to work with a CUDA-aware version of MPI [6].

- If we are operating on **sparse graph data sets**, we would have to look into sparse matrix representations and tune our operations to perform well on these sparse representations. If data partitioning is not implemented correctly, we could potentially run into load balancing and gang skew issues as well.

4 Resources

We will be using a couple of papers to help us get familiar with the mathematical theory of spectral clustering, as well as sequential implementations of the same. We would also be referring to previous (non-hybrid) parallel implementations of parts of the algorithm, such as graph construction and eigen value computation. We have outlined these below.

- A CUDA-only parallel implementation of the Lanczos algorithm [7].

- Mathematical theory of variants of the Lanczos algorithm [5].

- A tutorial on the mathematical theory of spectral clustering [2].

- More papers on the mathematics of Spectral Clustering [4, 3].

- Approximate, fast Spectral Clustering algorithms [8, 9].

- Spectral Clustering for community detection in large graphs [10].

- Using CUDA to accelerate spectral graph analysis [11].

- An introduction to CUDA-Aware MPI [6].

- An efficient serial MATLAB implementation of Spectral Graph Clustering [12].
- A Scikit-lean Python implementation of Spectral Clustering [13].
- A parallel K-Means clustering algorithm with MPI [14].
- A paper on Spectral Clustering in distributed systems [15].
- The Stanford Network Analysis Project [1].

5 Goals and Deliverables

5.1 Plan to Achieve

We will have a working, correct implementation of a C++ hybrid MPI + CUDA parallel version of a Spectral Clustering algorithm as a miniature library. We expect the quality of the results produced by our algorithm to be comparable [1] to those produced by a sequential version of the algorithm.

The previous CUDA-only implementation [7] achieved **5x** speedup over a sequential MATLAB version. Pessimistically, we expect to achieve at least **5x** speedup over the sequential MATLAB version as well, but on larger data sets (as we have more total memory and compute power available to us).

We will also compare our code with MPI-only (if available) and CUDA-only parallel implementations of the algorithm, and hope to stay competitive. We anticipate two major bottlenecks in our code: inter-node communication overhead in MPI and CUDA host memory to device memory transfer overhead. We will spend a significant amount of time optimizing and analyzing these two.

For the poster session, we will show a visualization of our speedup graphs, and discuss the parallelization approaches and trade-offs that we made. We will also show the interface to the library, the API structure and the easy of use of our code.

5.2 Hope to Achieve

We hope to achieve at least **(5 + 0.5 * number of nodes)x** speedup on two data sets. The reason is that we expect the algorithm to scale with the number of nodes as well, providing more speedup than the **5x** speedup obtained using the CUDA-only implementation. However, we anticipate non-ideal speedup and limit ourselves to a more achievable goal of 0.5 * number of nodes.

Further, if our project goes better than expected, we will test against a few more data sets (with varying degrees of sparsity) and study the scaling characteristics of our code on

[1] If we do decide to trade quality for speedup, we will make sure to document and bound the error in our algorithm.

these data sets. In the best case scenario, we would also perform isolated testing on AWS machines or on different GPUs to analyze the variability in our results.

5.3 In Case of Delays

Conversely, if our project does not go according to the time line due to unforeseen issues, we will implement a subpart of the algorithm completely and correctly, and describe exactly what extra work would need to be done to convert the partial results into the final expected result.

If our speedups are not as we expected, we will study the scaling properties of various parts of the algorithm individually and point out exactly why our algorithm does not work as we expected.

6 Platform Choice

- This project will be implemented using C++, CUDA and CUDA-aware MPI.
- Development will be done on local machines + GHC cluster machines. Debugging will also be done on the same.
- Our algorithm will be tested on the GHC cluster machines with GTX 1080s and OpenMPI. Our algorithm will not make any assumptions about the interconnect, but we will study time spent on communication between nodes.
- We may choose to use AWS machines or the Latedays cluster for isolated testing, if the variance in the obtained results is too high.

Our code will be written from scratch in C++, but may choose to use Thrust/cuBLAS helper routines. We may study ideas from the previous CUDA-only versions to understand how they fit into our MPI + CUDA formulation of the algorithm, and decide to use a few of them (while providing due credit).

7 Schedule

Start Date	End Date	Work to be done
Nov 1	Nov 5	Understand the mathematics behind Spectral Clustering. Finalize an eigen decomposition algorithm. Finalize on a test data set. Compute benchmarks using existing implementations.
Nov 6	Nov 13	Implement sequential Spectral Clustering. Compare against benchmarks. * *Light workload week as we each have multiple tests.*
Nov 14	Nov 20	Prepare the hybrid MPI + CUDA architecture. Develop subroutines for Spectral Clustering using CUDA. Distribute tasks across OpenMPI processes. Use small data sets to run correctness tests.
Nov 21	Nov 27	Optimize the algorithm for large data sets. Work on Minimizing MPI/CUDA overheads. Run benchmark tests on large data sets.
Nov 28	Dec 4	Buffer for pending tasks/running tests.
Dec 5	Dec 12	Write the final project report. Prepare poster for presentation. Create necessary pending visualizations.

References

[1] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[2] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[3] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[4] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances In Neural Information Processing Systems*, pages 849–856. MIT Press, 2001.

[5] Jane Cullum and Ralph A. Willoughby. A survey of lanczos procedures for very large real 'symmetric' eigenvalue problems. *Journal of Computational and Applied Mathematics*, 12:37 – 60, 1985.

[6] Jiri Kraus. An introduction to cuda-aware mpi. `https://devblogs.nvidia.com/parallelforall/introduction-cuda-aware-mpi/`. Published: 2013-03-13.

[7] Parallel eigensolver for graph spectral analysis on gpu. `http://linhr.me/15618/`. Accessed: 2016-10-31.

[8] Xinlei Chen and Deng Cai. Large scale spectral clustering with landmark-based representation. In *AAAI*, 2011.

[9] Donghui Yan, Ling Huang, and Michael I. Jordan. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 907–916, New York, NY, USA, 2009. ACM.

[10] Scott White and Padhraic Smyth. *A Spectral Clustering Approach To Finding Communities in Graphs*, pages 274–285.

[11] Maxim Naumov. Fast spectral graph partitioning on gpus. Nvidia Developer Blog: `https://devblogs.nvidia.com/parallelforall/fast-spectral-graph-partitioning-gpus/`. Published: 2016-05-12.

[12] Fast and efficient spectral clustering in matlab. Mathworks Tutorial Article: `https://www.mathworks.com/matlabcentral/fileexchange/34412-fast-and-efficient-spectral-clustering`. Accessed: 2016-10-31.

[13] Clustering using scikit-learn. `http://scikit-learn.org/stable/modules/clustering.html`. Accessed: 2016-10-31.

[14] J. Zhang, G. Wu, X. Hu, S. Li, and S. Hao. A parallel k-means clustering algorithm with mpi. In *2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming*, pages 60–64, Dec 2011.

[15] W. Y. Chen, Y. Song, H. Bai, C. J. Lin, and E. Y. Chang. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):568–586, March 2011.