Multi-GPU Implementation of Bisection Algorithm for Symmetric Tridiagonal Eigenvalue Problem

Barok Imana, Nam Thai and Peter Yoon

Department of Computer Science, Trinity College, Hartford, CT

Abstract

Bisection is a numerically stable algorithm used to find the eigenvalues of **symmetric tridiagonal matrices**. It is distinct from other methods used to solve the eigenvalue problem by the high precision of its results. However, on a sequential processor, the algorithm is significantly slow compared to other methods. But the algorithm is suitable for parallel processors because of its highly parallelizable features. We have exploited these features to implement the bisection algorithm on a single GPU. We also extended the algorithm to multi-GPU platform. Our implementation, on average, runs 30 % faster on 2 GPUs than on a single GPU. [1]

Bisection Algorithm

i. Gerschgorin Interval

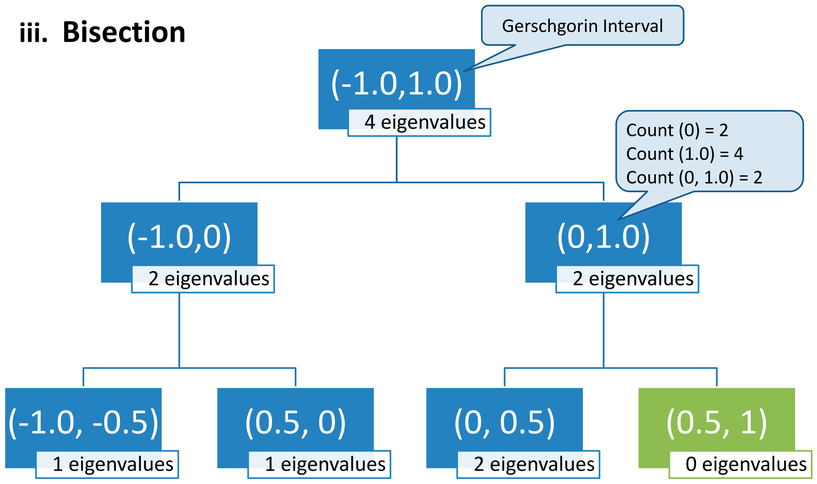
- The Gerschgorin interval gives the bounds for the eigenvalue spectrum of a given square matrix.

ii. *Count(x)* function

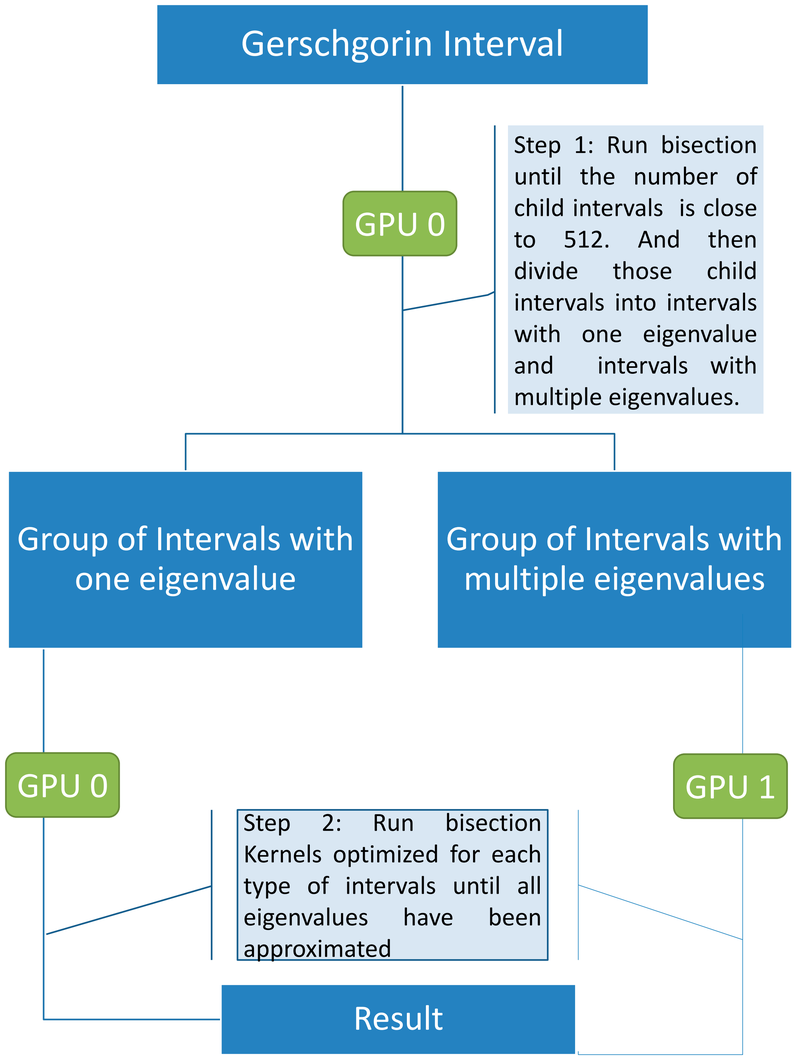
- The *Count(x)* function gives the number of eigenvalues less than the number x. It can be used to find the number of eigenvalues within an interval (x, y]as follows:

  Count(x, y) = Count (y) – Count (x)

iii. Bisection



Multi-GPU based Approach



Acknowledgement

This research was supported by:

- CUDA Teaching Center Program, NIVIDIA™ Research
- Summer Research Program, Trinity College

Preliminary Results

Matrix size	Average running time (sec)			Speed-up (16 CPU cores to 2 GPUs)	Speed-up (2 GPUs and 1 GPU)
	16 CPU cores	1 GPU	2 GPU		
1024	0.114	0.082	0.056	2.0	1.46
2048	0.576	0.156	0.100	5.8	1.56
4096	2.19	0.316	0.194	11.3	1.63
8192	8.90	0.850	0.590	15.1	1.44
16384	35.9	2.210	1.67	21.5	1.32
32768	138	7.360	6.27	22.0	1.17

- Testing environment
  - Intel™ Xeon™ E5-2620 CPUs, 64 GB
  - NVIDIA™ Tesla™ K20c GPUs, 5 GB
- An average of 30 % speed up on 2 GPUs over single GPU

Future Work

We are considering two other approaches for the multi-GPU algorithm

i. Evenly dividing the Gerschgorin interval among the GPUs
   - This makes each GPU independent and reduces GPU-GPU communication

ii. At the end of step one, assigning the intervals to each GPU so that each GPU will have to find the same number of eigenvalues
   - This will lead to an even workload among the GPUs

Reference

[1] Lessig, Christian. "Eigenvalue Computation with CUDA." Oct. 2007.