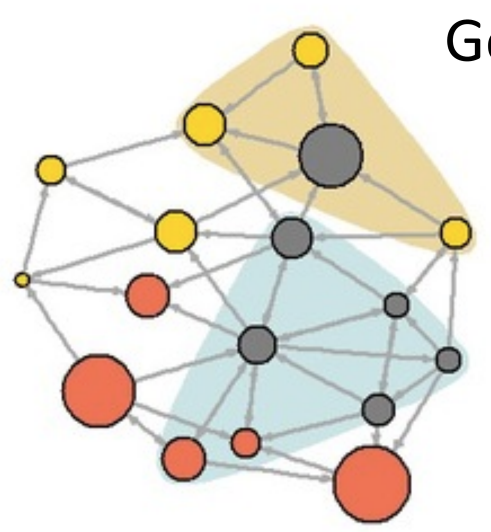
GPU Implementation of Triangle Counting Algorithm without Matrix Multiplication

Minji Kim | Supervisor: Franz Franchetti, Tze Meng Low

Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA

Background



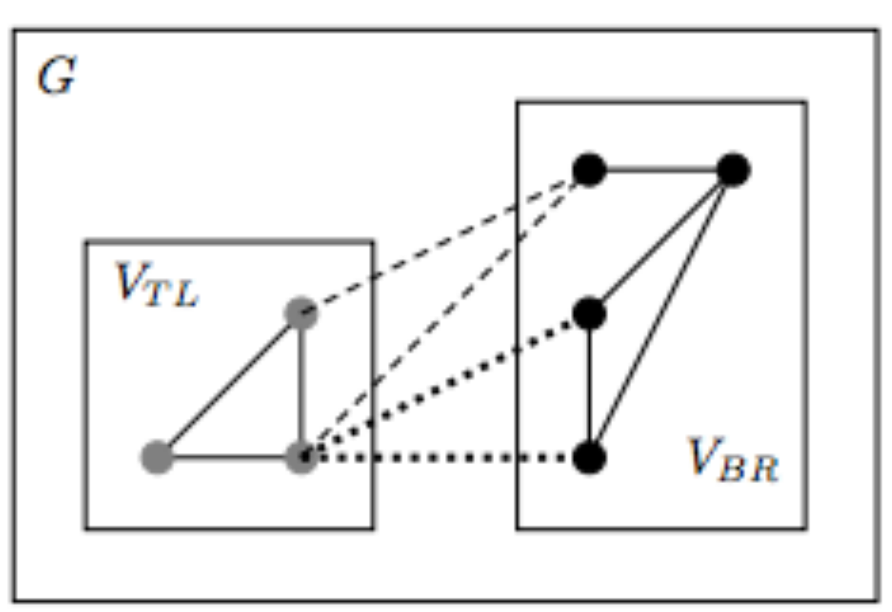
General Equation for counting triangles in a graph:

$$\frac{1}{6}\,\Gamma(A^3)$$

How do you increase the efficiency of counting triangles for large graphs?

Algorithm

Linear-Algebra Based Algorithm without Matrix Multiplication



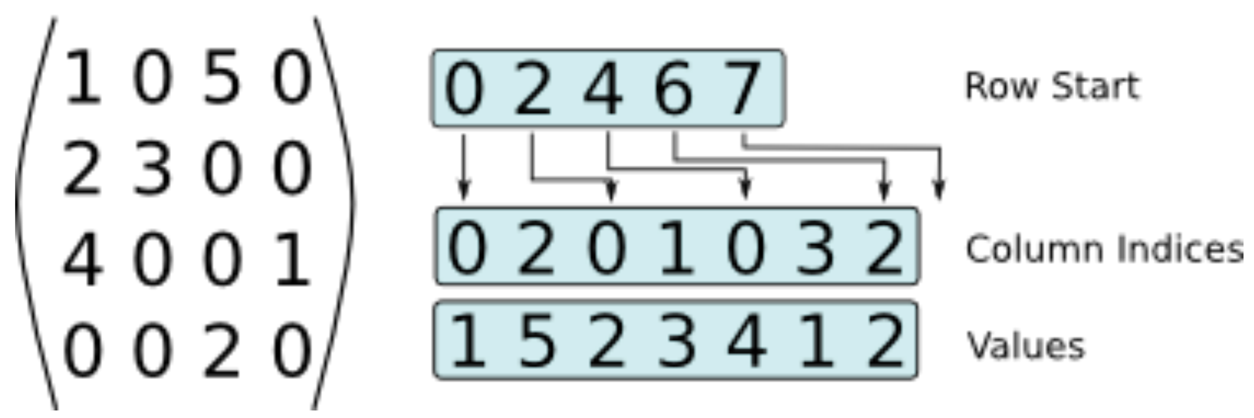
- Place all vertices in V_{BR}
- Move an arbitrary vertex v to V_{TL}
- Add the number triangles where v was the last vertex remaining in V_{BR} to delta
- Repeat the process until all vertices are placed in V_{TL}

Linear Algebra Implementation

$$\left(\begin{array}{c|c} A_{00} & (a_{01}|A_{02}) \\ \hline \begin{pmatrix} a_{10}^T \\ \hline A_{20} \end{pmatrix} & \begin{pmatrix} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{pmatrix} \end{array}\right) \rightarrow \left(\begin{array}{c|c} \begin{pmatrix} A_{00} & a_{01} \\ \hline a_{10}^T & \alpha_{11} \end{pmatrix} & \begin{pmatrix} A_{02} \\ \hline a_{12}^T \end{pmatrix} \\ \hline (A_{20}|a_{21}) & A_{22} \end{array}\right)$$

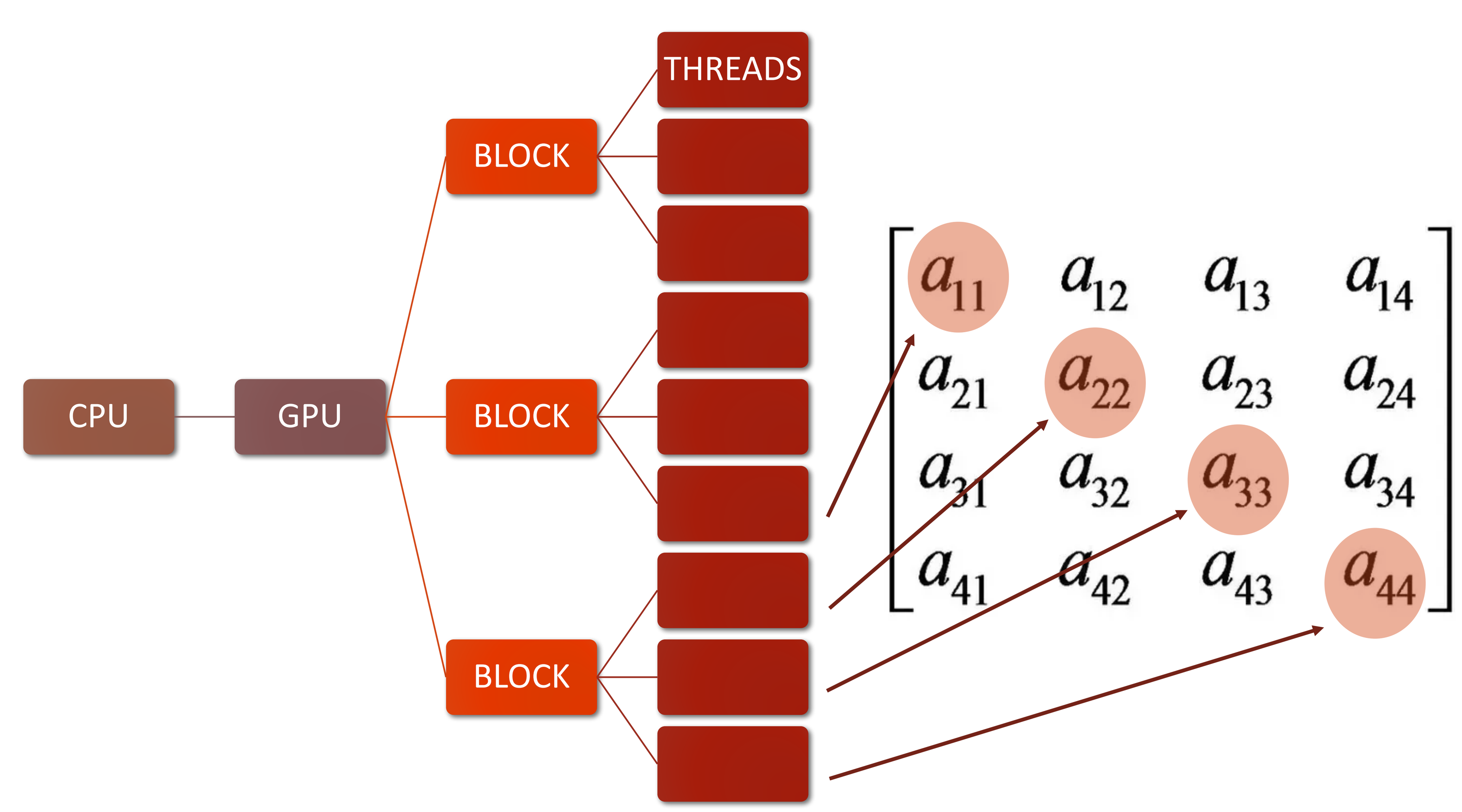
$$\Delta = \Delta + a_{12}^T A_{20} a_{01}$$

The Adjacency Matrix is stored in **CSR Format**



GPU Implementation

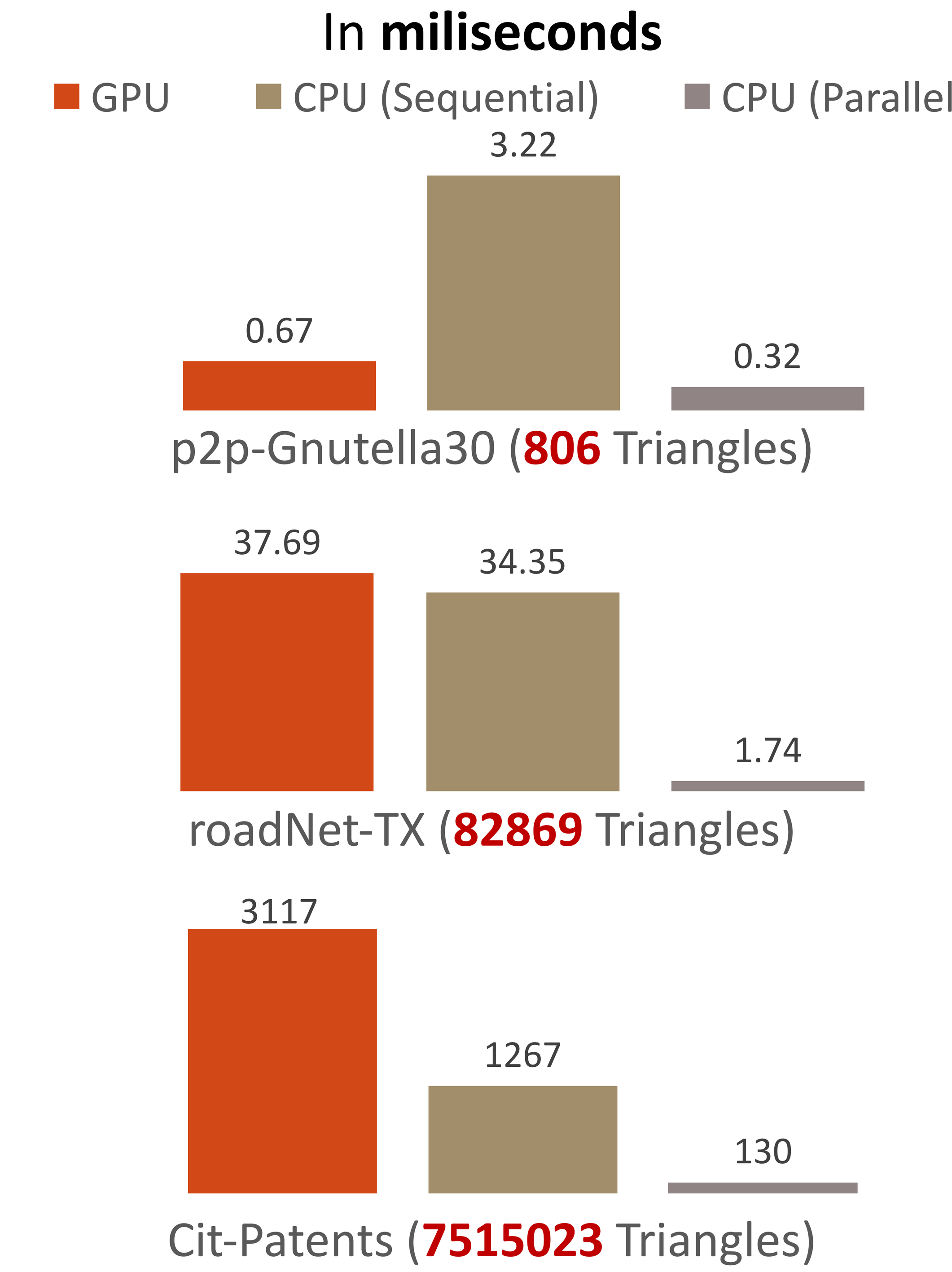
CUDA NVIDIA CUDA + NUMBA



Preventing race condition when updating delta:

```
cuda.atomic.add(delta,0,1)
```

Performance Comparison



Conclusion

- As the size of the graph and the number of the triangles increase, the kernel execution time increase exponentially.
- Thread synchronization time required during GPU operation due to varying loop sizes may have caused the GPU implementation to take longer than CPU parallel implementation.
- CUDA Numba also requires the Numba module to compile the kernel in C language, which may have affected the execution time significantly.

Future Plans

- Further optimization of current parallelism algorithm
- GPU Implementation using CUDA C and comparing results