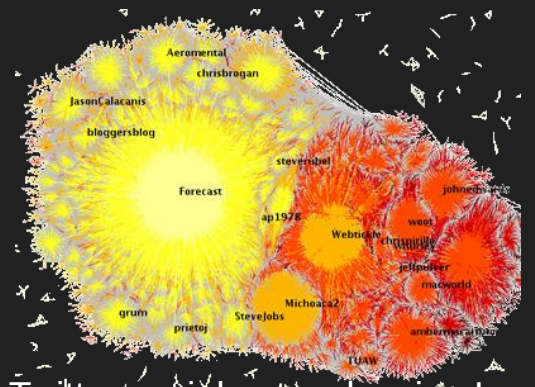
Fast Triangle Counting on the GPU

Oded Green

Triangle Counting

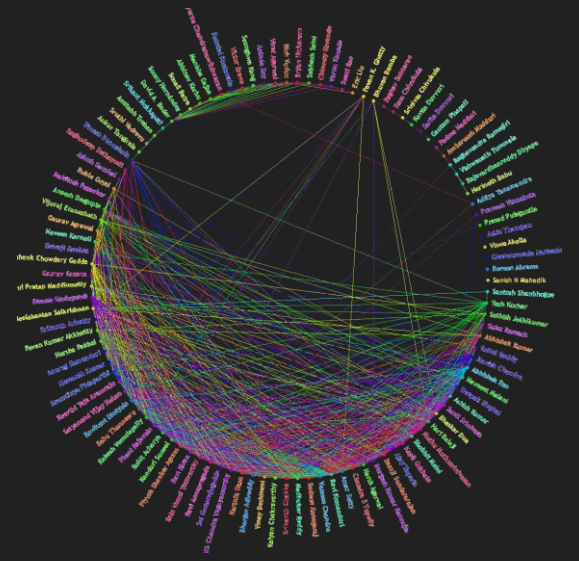
- Building block for clustering coefficients
  - Defined by **[Watts & Strogatz; 1998]**
  - Used to state how tightly bound vertices are in a network
- Used for the analysis many types of networks:
  - Communication
  - Social
  - Biological

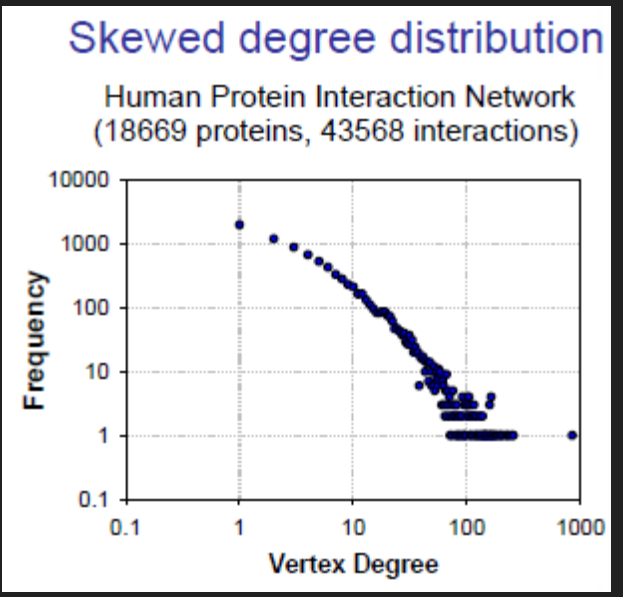


Twitter social network using Large Graph Layout

Relevant network properties

- Sparse-network
  - System utilization
- Power-law distribution
  - **[Faloutsos & Faloutsos & Faloutsos; 1999]**
  - **[Broder *et al.*; 2000]**
  - Load balancing issues



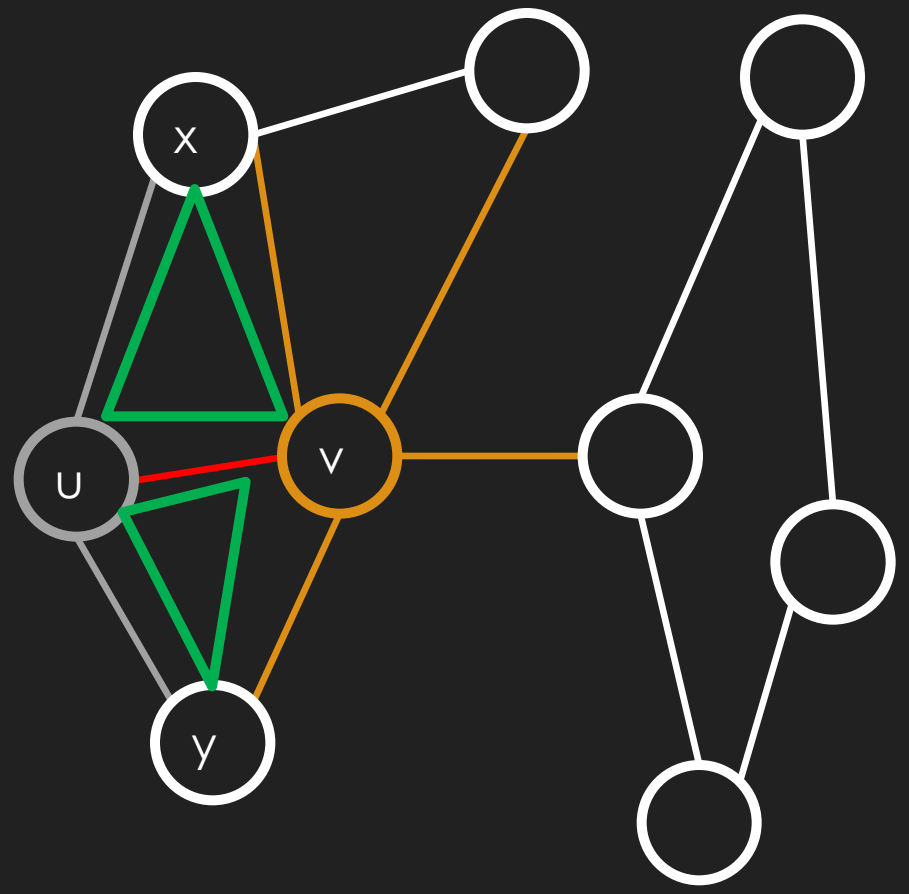


Computational Approaches

- Enumerating over all node-triples - $O(V^3)$.
- Using matrix multiplication - $O(V^w)$, $w \leq 2.376$.
- Adjacency list intersection - $O(V \cdot d_{max}^2)$, where d_{max} is the vertex with largest adjacency.

Adjacency List Intersection

- For given e = $(u, v) \in E$:
  - Take:
    - u's list
    - v's list
  - Intersection and find common neighbors
- Then intersect (u, x) and (u, y)

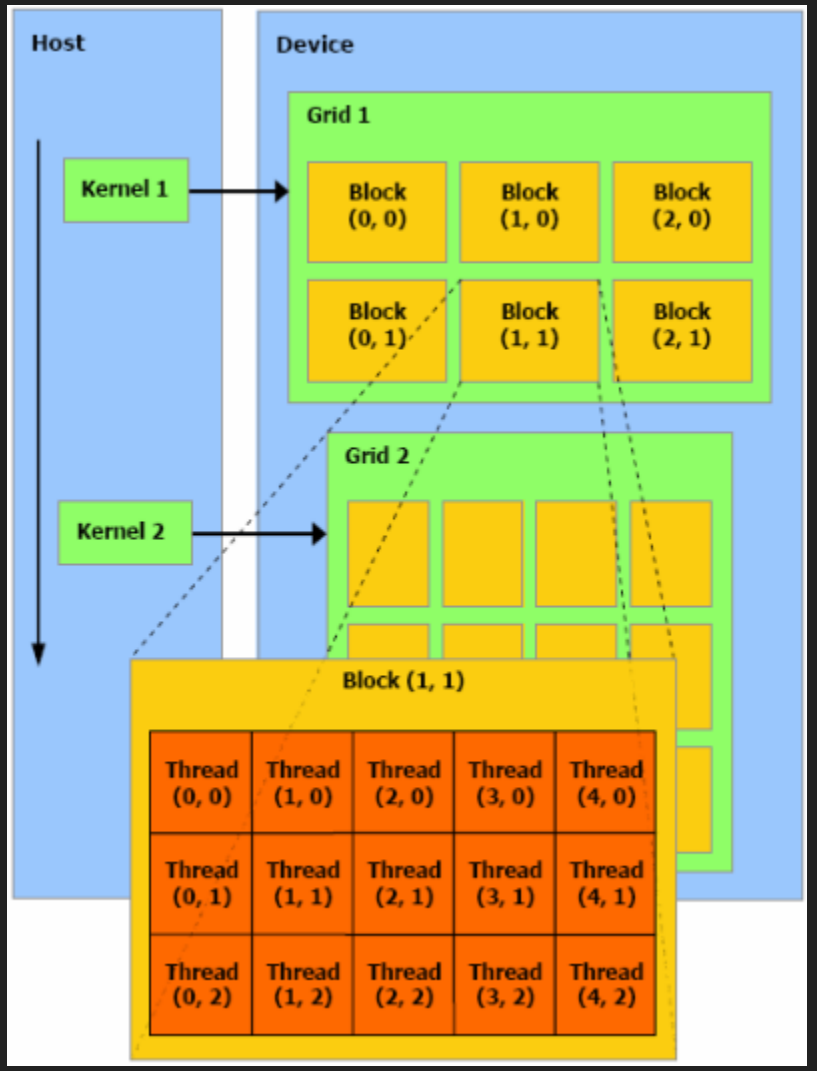


GPU Challenges: List Intersection

- Partitioning / Load balancing
  - Must be computationally cheap
  - Must be parallel (high utilization)
- Minimal synchronization/communication
- Scalable
- Simple

The only other GPU triangle counting algorithm

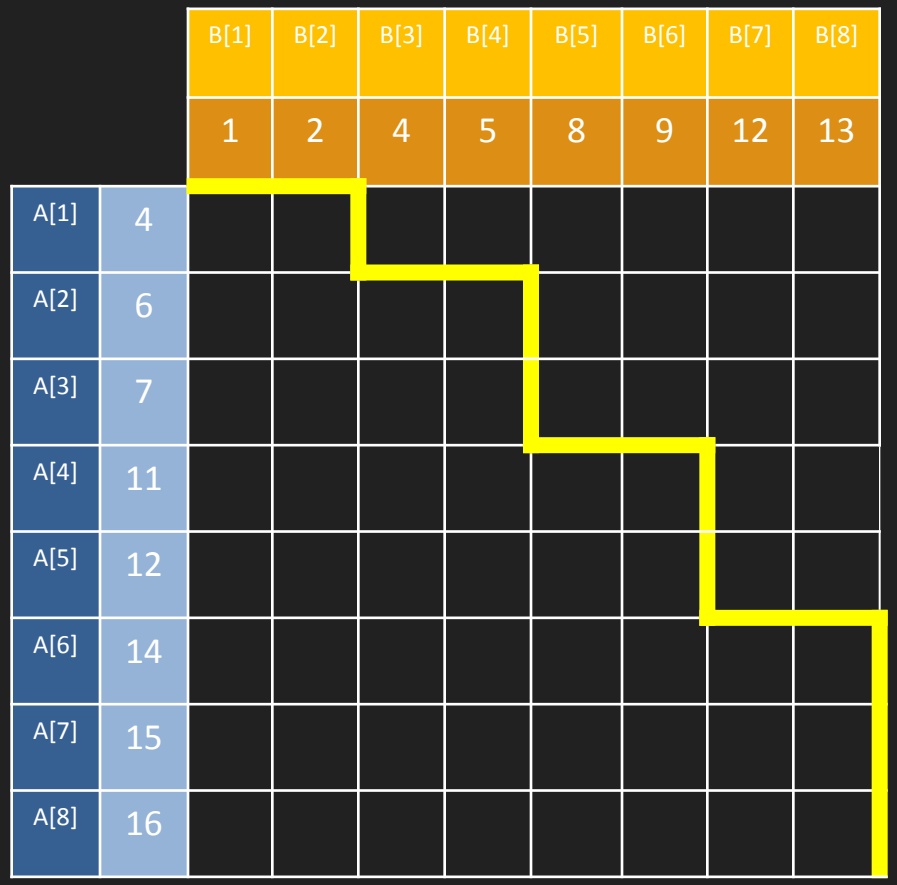
- Uses the GPU like a CPU
- One CUDA thread per intersection
  - Load balancing issues
  - Low utilization
  - Limited scalability
- **[Heist *et al.* ;2012]**



Merge-Path and GPU Triangle Counting

oded@arrayfire.com, GTC, 2015

Merge-Path

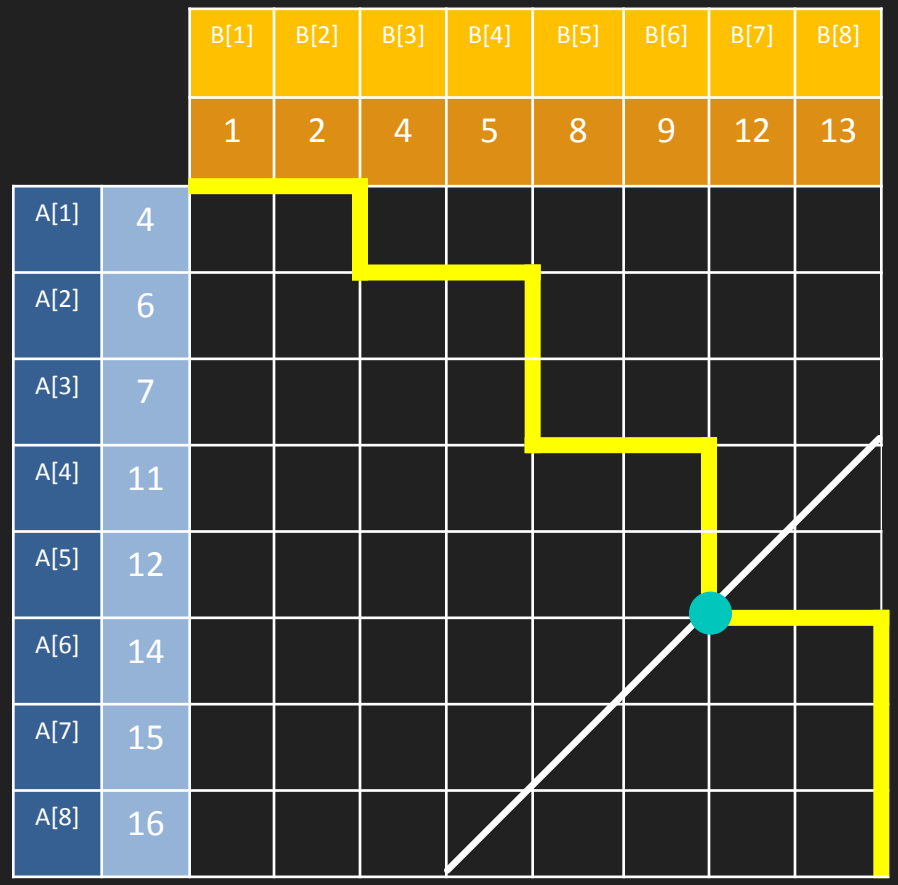


- Visual approach for merging
- Highly scalable
- Load-balanced
- Two legal moves
  - Right
  - Down
- Reminder: Merging and intersecting are similar

Merge Path : **[Odeh et al. ; 2012]**
GPU Merge Path : **[Green et al. ;2012]**

Partitioning

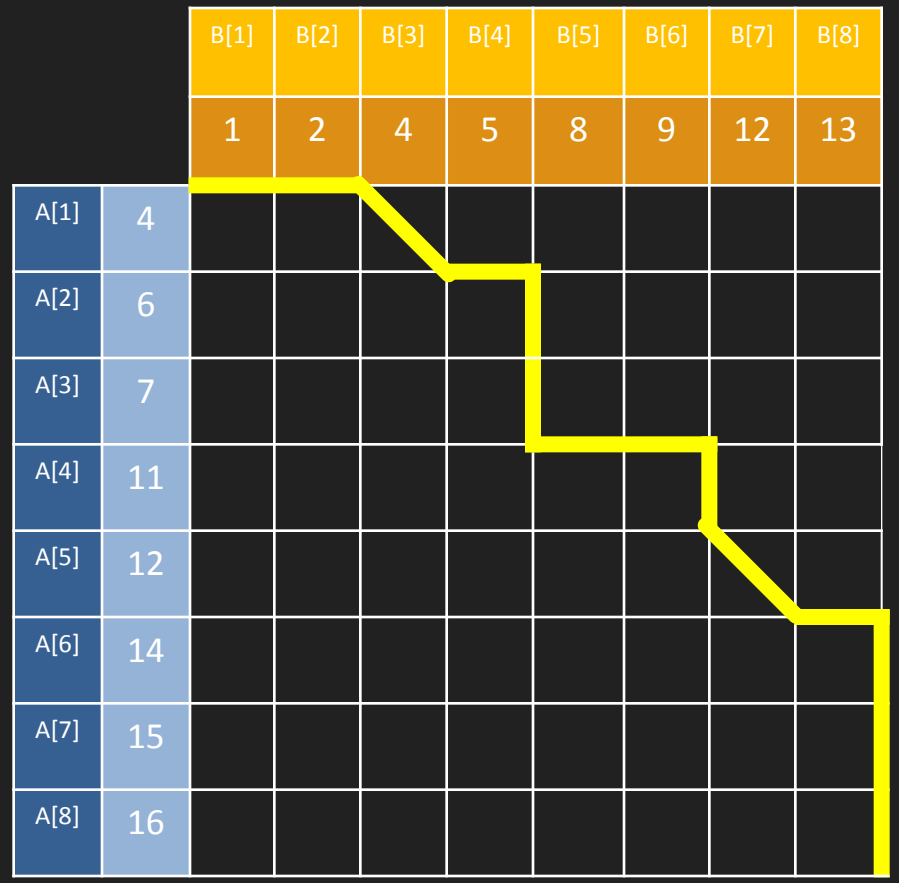
- Find the intersection of the cross-diagonal and the path
  - Uses binary search



Merge-Path Phases

- Partitioning
  - Log(n)
- Synchronization
- Merging
- Synchronization

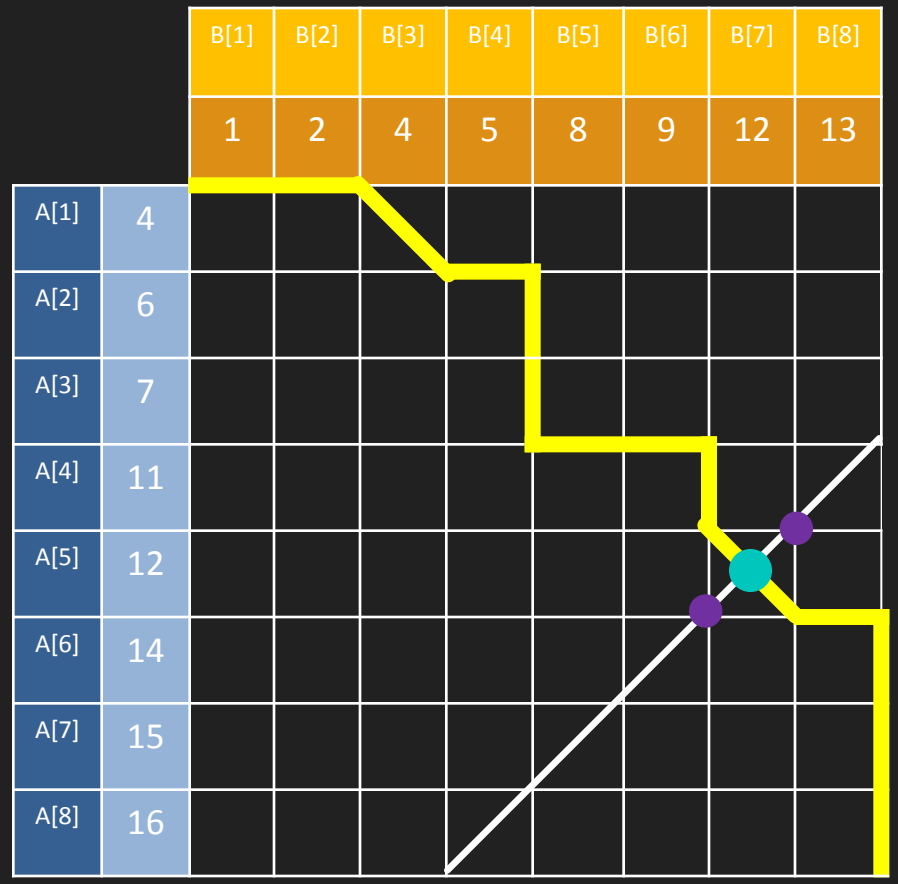
Intersect-Path



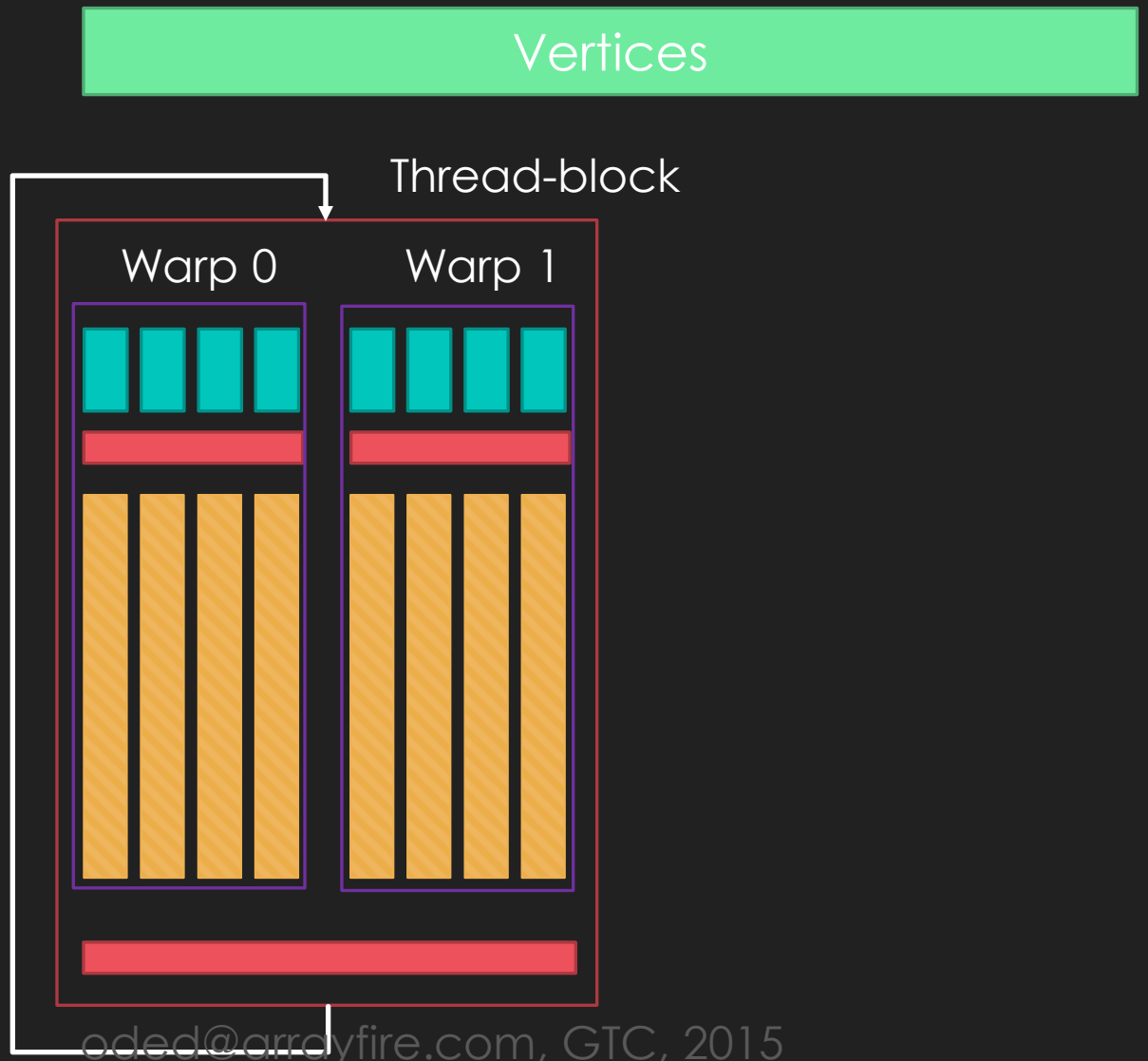
- Building block for triangle counting and clustering coefficients
- Modified Merge-Path
  - Right
  - Down
  - Right-down – when values are equal

Differences

- Typically unique values in each set
- Two possible start locations when cross-diagonal goes through “equal-path”.
- Requires:
  - Checking the initial condition
  - Avoiding double counting
  - Shared-memory and synchronization



Triangle-Counting



- Split vertices to thread blocks
  - |V|/Thread_Blocks
- Iteratively (over the vertices)
  - Partition
    - Log(n)
  - Warp Synchronization
  - Multiple Intersection
  - Thread-block Synchronization

Algorithm Control Parameters

- I: # intersection in a thread block
  - Small number : under utilization
  - Large number : not useful for sparse networks
- Th: # threads per intersection
  - Small number : load-balancing, under utilization
  - Large number : adds overhead compared to the intersection

$$I \cdot Th \geq warp\ size$$

Performance Analysis

Experiments

- Graphs taken from 10th DIMACS Challenge

- GPU
  - NVIDIA K40
  - 12 GB
  - 14 MPs x 192 SPs (per MP) = 2880 CUDA threads

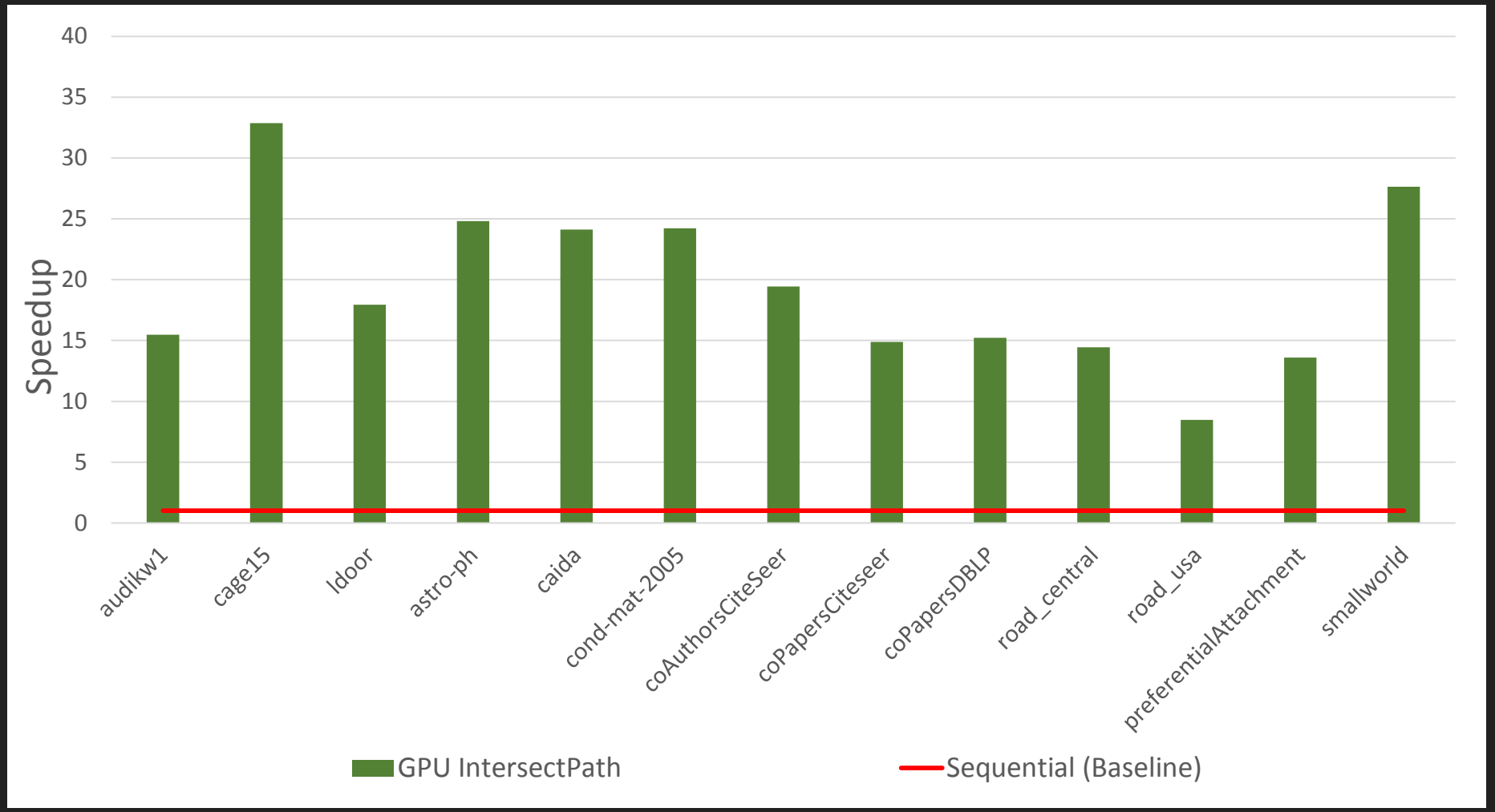
Experiments

- CPU - OpenMP based **[Green et al. ;2014]**
  - 4x10 Intel Xeon E7-8870 processers
  - 256 GB
  - 30 MB L3 cache per processor

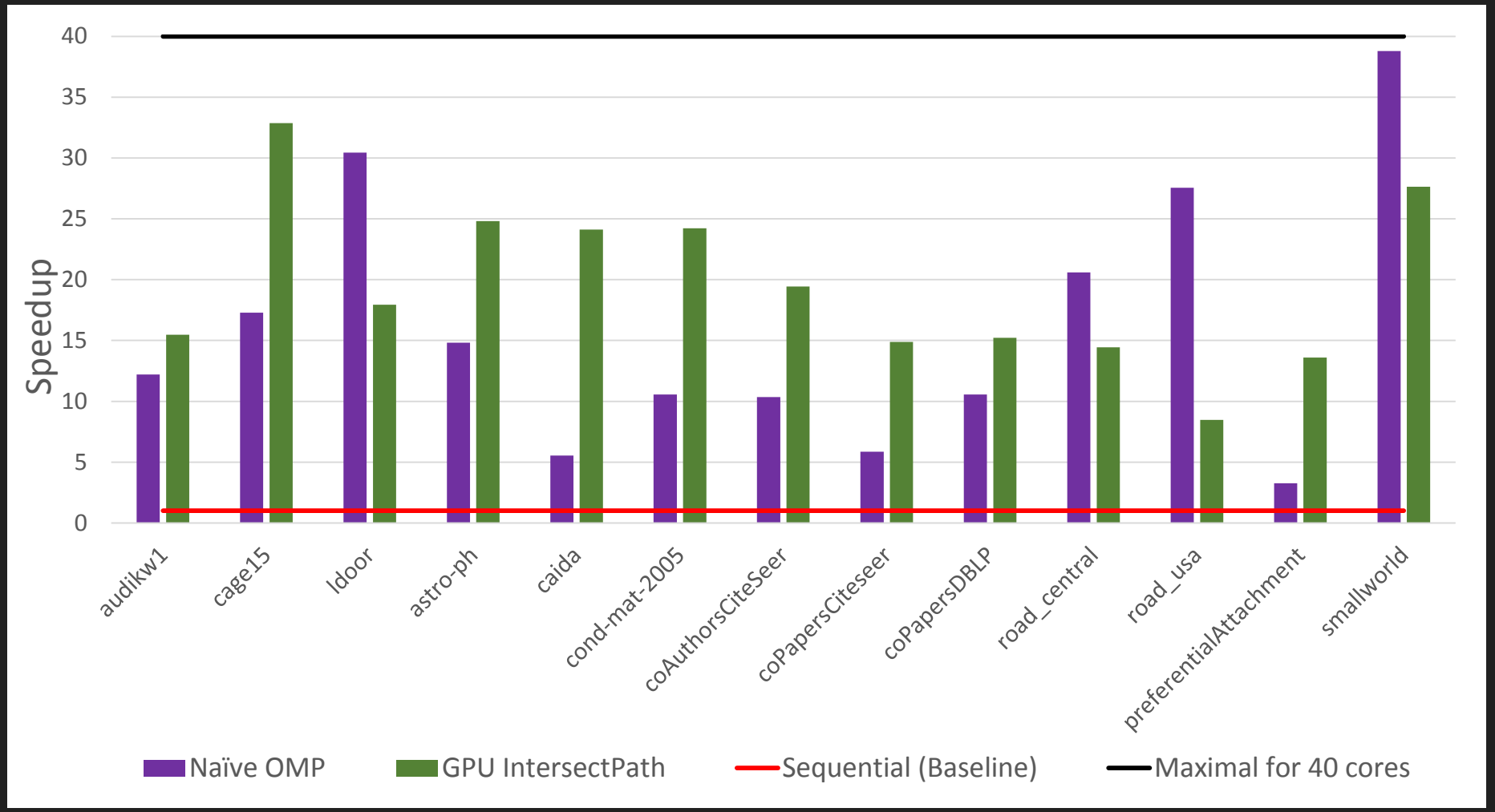
Implementation comparison

- GPU algorithm as discussed here
- CPU algorithms
  - Threads are independent
  - Partitioning stage not needed

GPU Vs. Sequential CPU

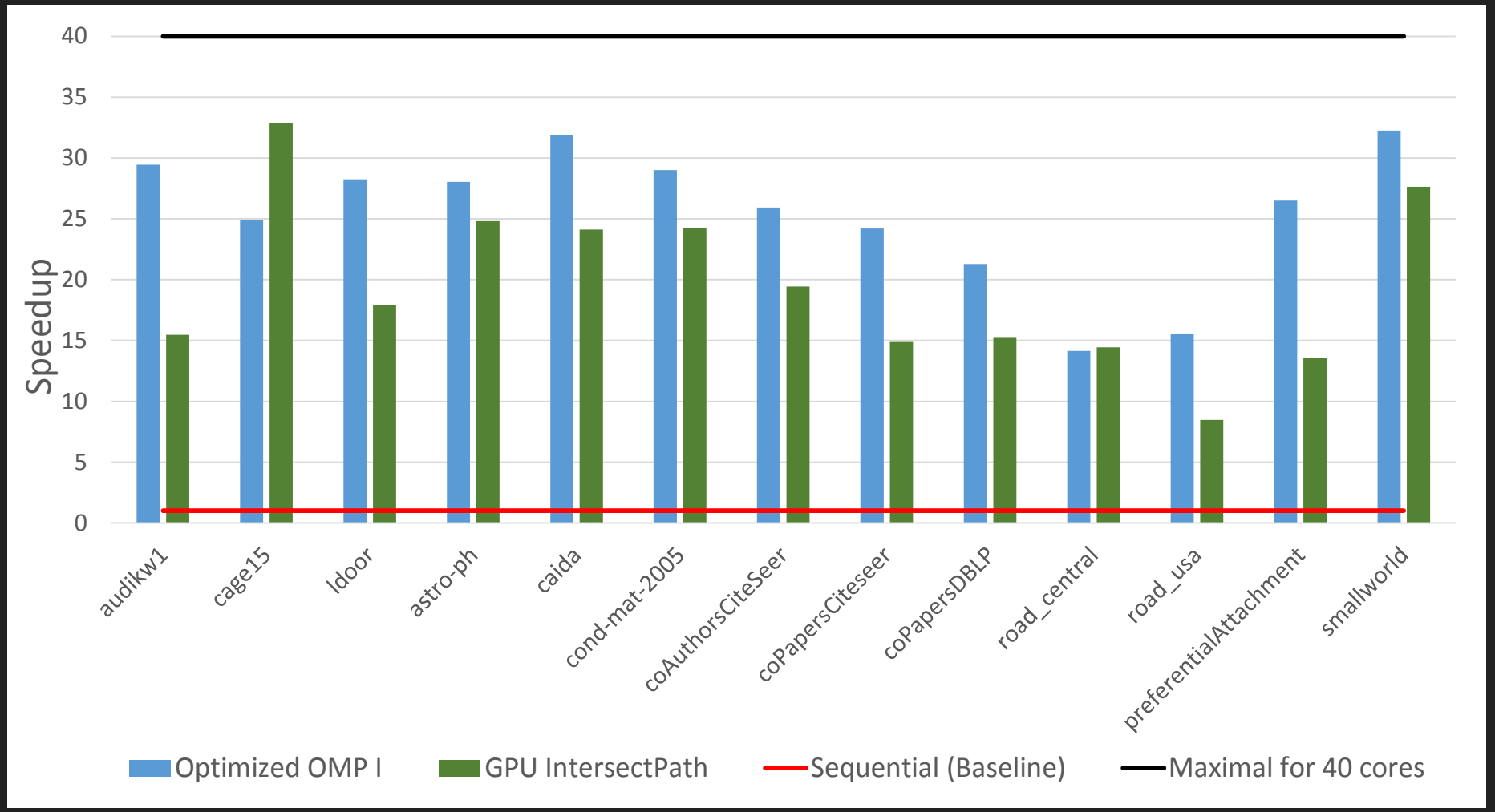


GPU Vs. CPU – OpenMP Straightforward



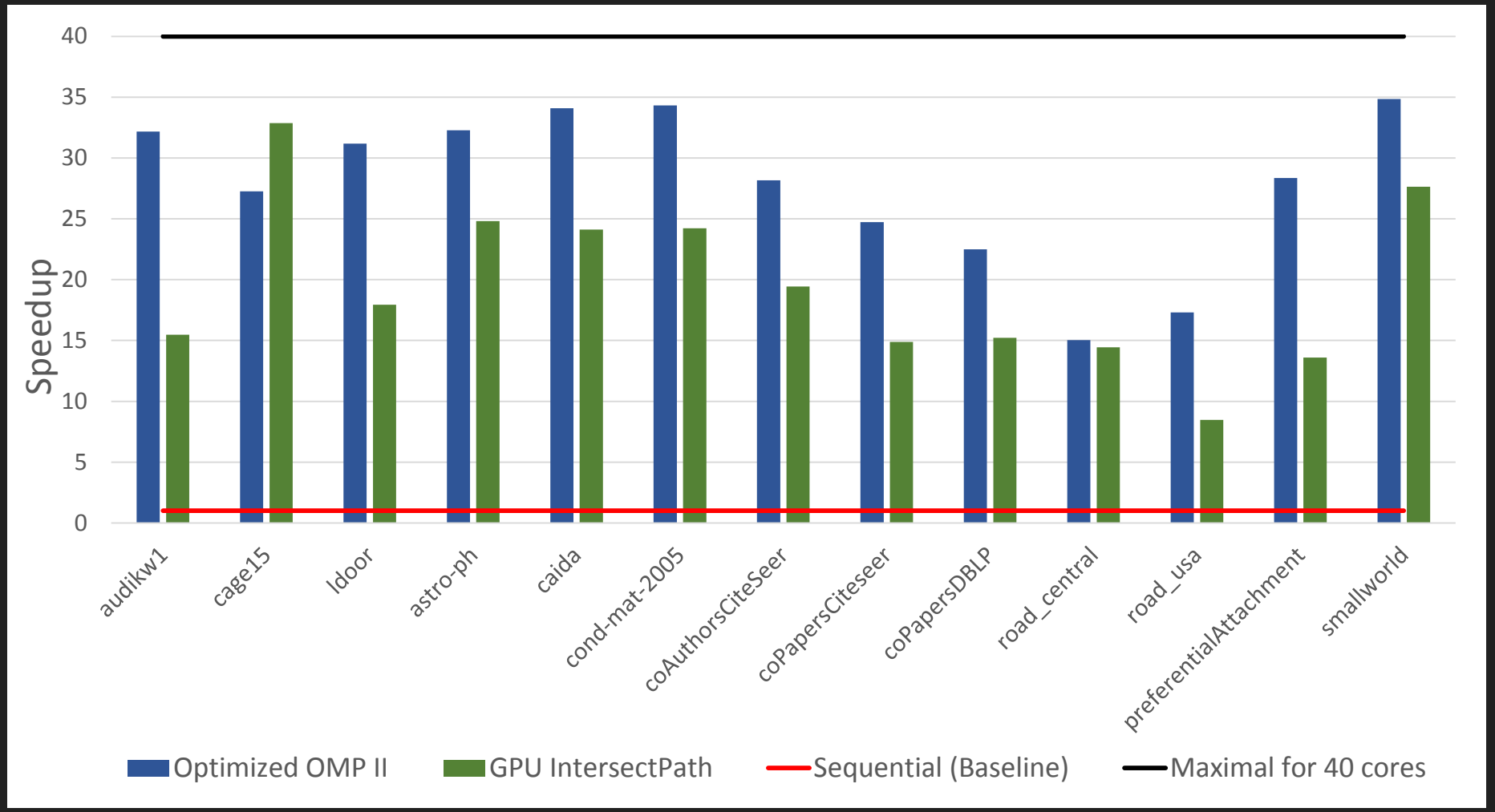
OpenMP results from[Green et al. ;2014]

GPU Vs. CPU OpenMP Optimized I



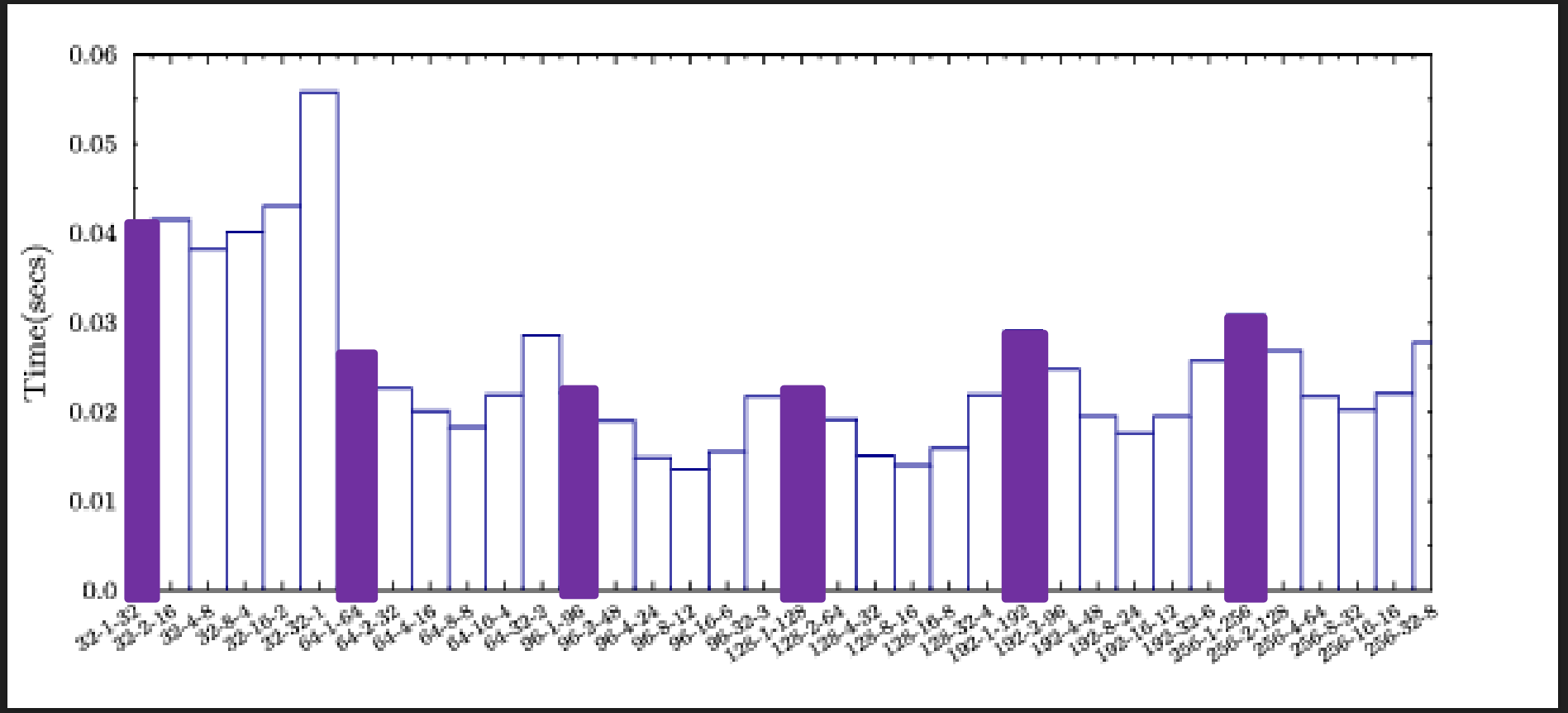
[Green et al. ;2014]

GPU Vs. CPU OpenMP Optimized II



[Green et al. ;2014]

GPU – Parameter control -prefAttachment

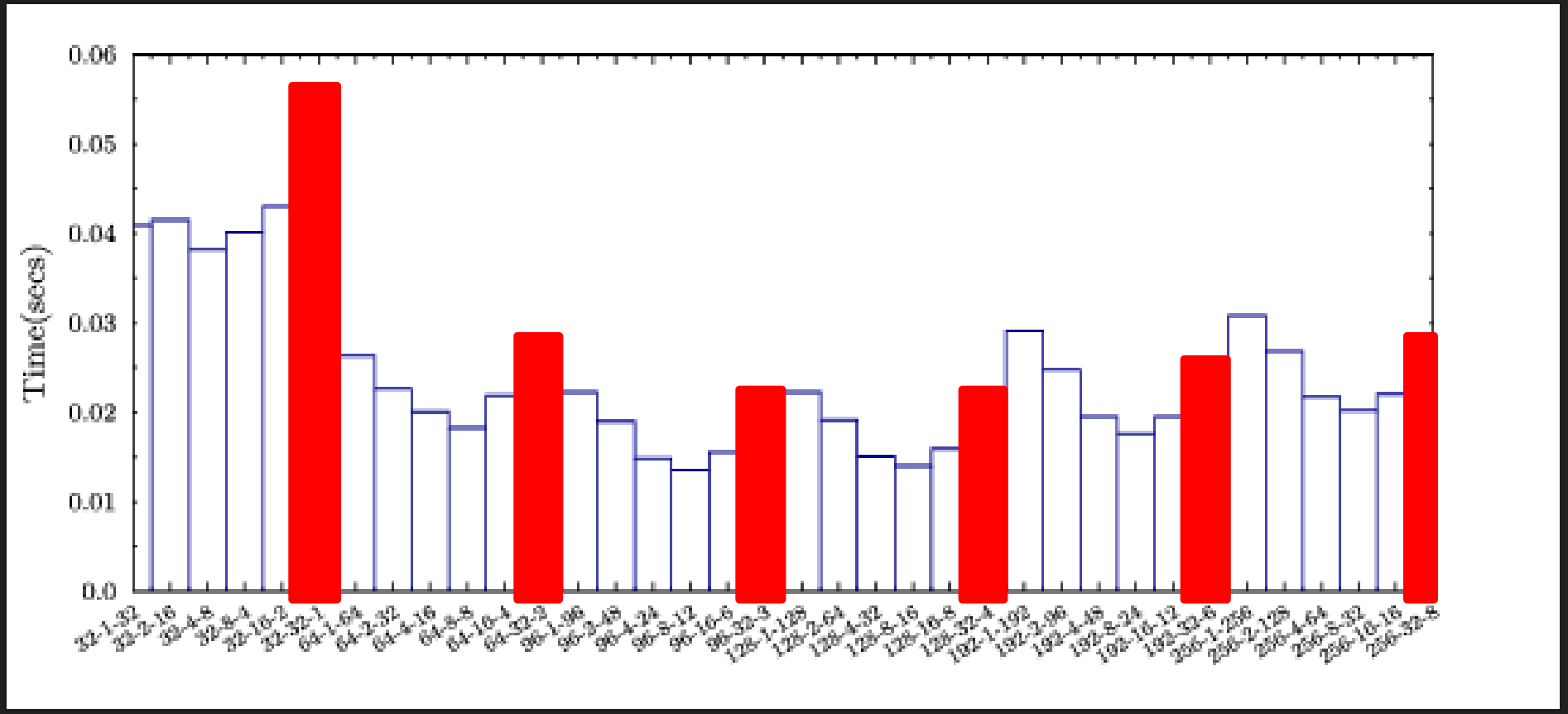


X-Y-Z configuration:

- X: threads per block
- Y: thread per intersection
- Z: number of intersection per thread block

Single thread per intersection

GPU – Parameter control -prefAttachment



X-Y-Z configuration:

- X: threads per block
- Y: thread per intersection
- Z: number of intersection per thread block

Single Intersection per warp

These parameters count

- $3X - 7X$ difference between slowest and fastest
- Offer better system utilization
- Offer better load-balancing

Final thoughts

- Intersect-Path
  - Modified Merge-Path
- Performance
  - 9X-32X faster than a sequential

Additional Challenges (Future work)

- Fine-tuning the GPU parameters based on graph properties.
- More load-balancing across the MPs.
- Create the capability to analyze large graphs on the GPU
  - Limited memory size

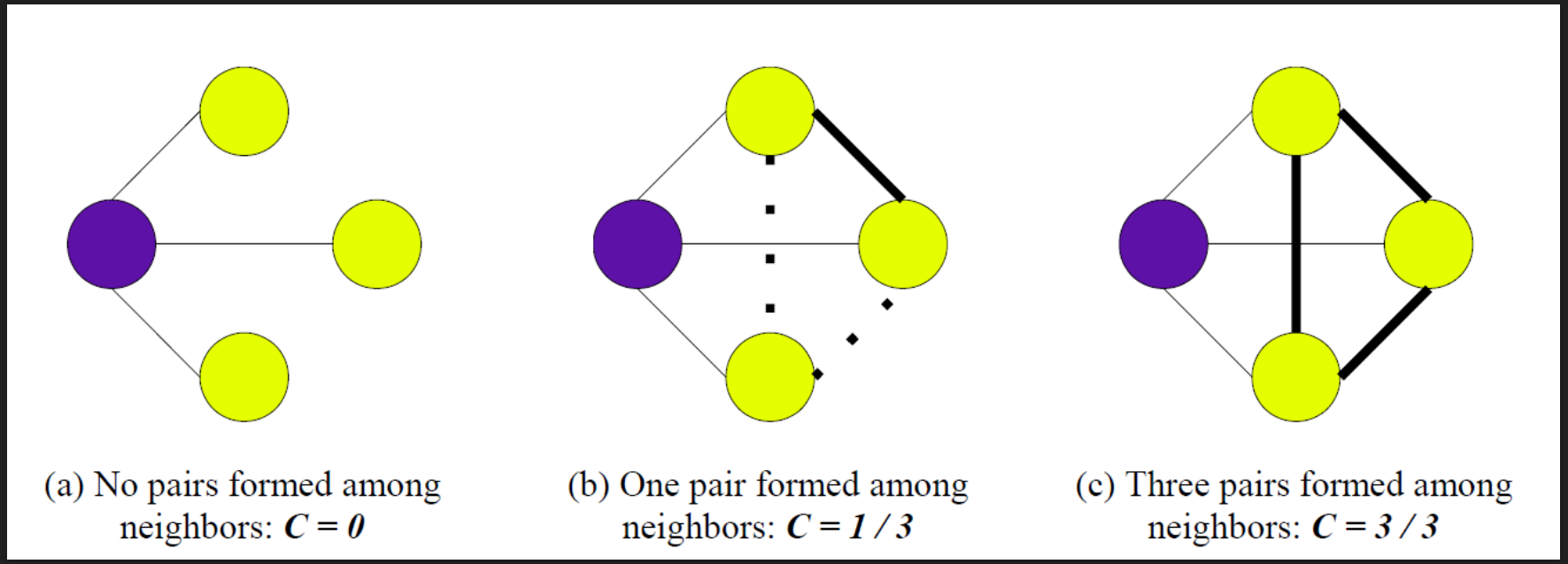
Thank you!

- Join our Open-Source community

https://github.com/arrayfire/arrayfire

Backup slides

Examples – Clustering Coefficient



(a) No pairs formed among neighbors: ***C = 0***

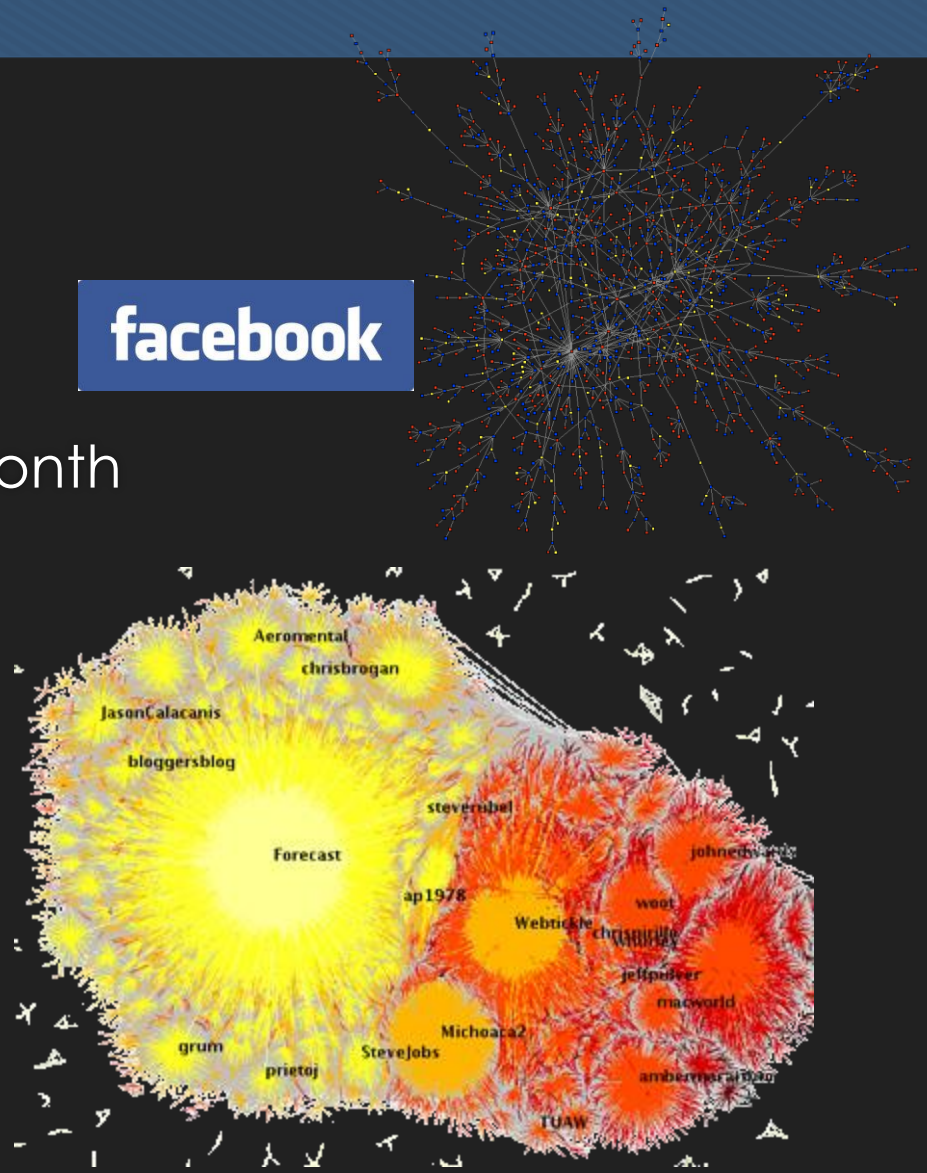
(b) One pair formed among neighbors: ***C = 1 / 3***

(c) Three pairs formed among neighbors: ***C = 3 / 3***

$$CC = \sum_{v} CC_v = \sum_{v} \frac{tri(v)}{\deg(v) \cdot (\deg(v) - 1)}$$

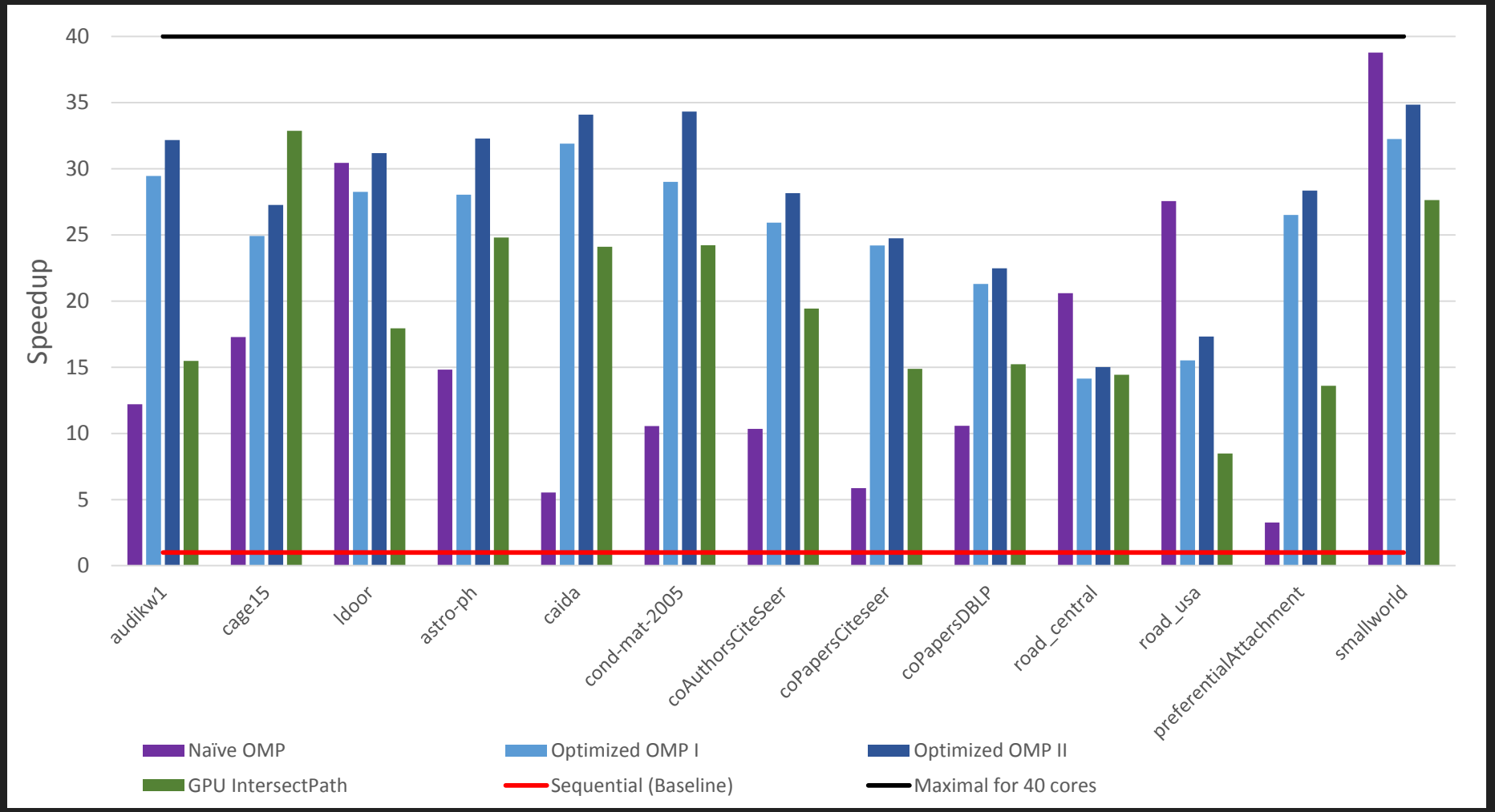
Motivation: Social Media

- Immense volume of data
- **Facebook**: >1B users
  - average 130 friends
  - 30B pieces of content shared / month
- **Twitter**: 200M active users
  - 400M tweets / day
- Goal: Use the interaction network to understand and characterize information flow

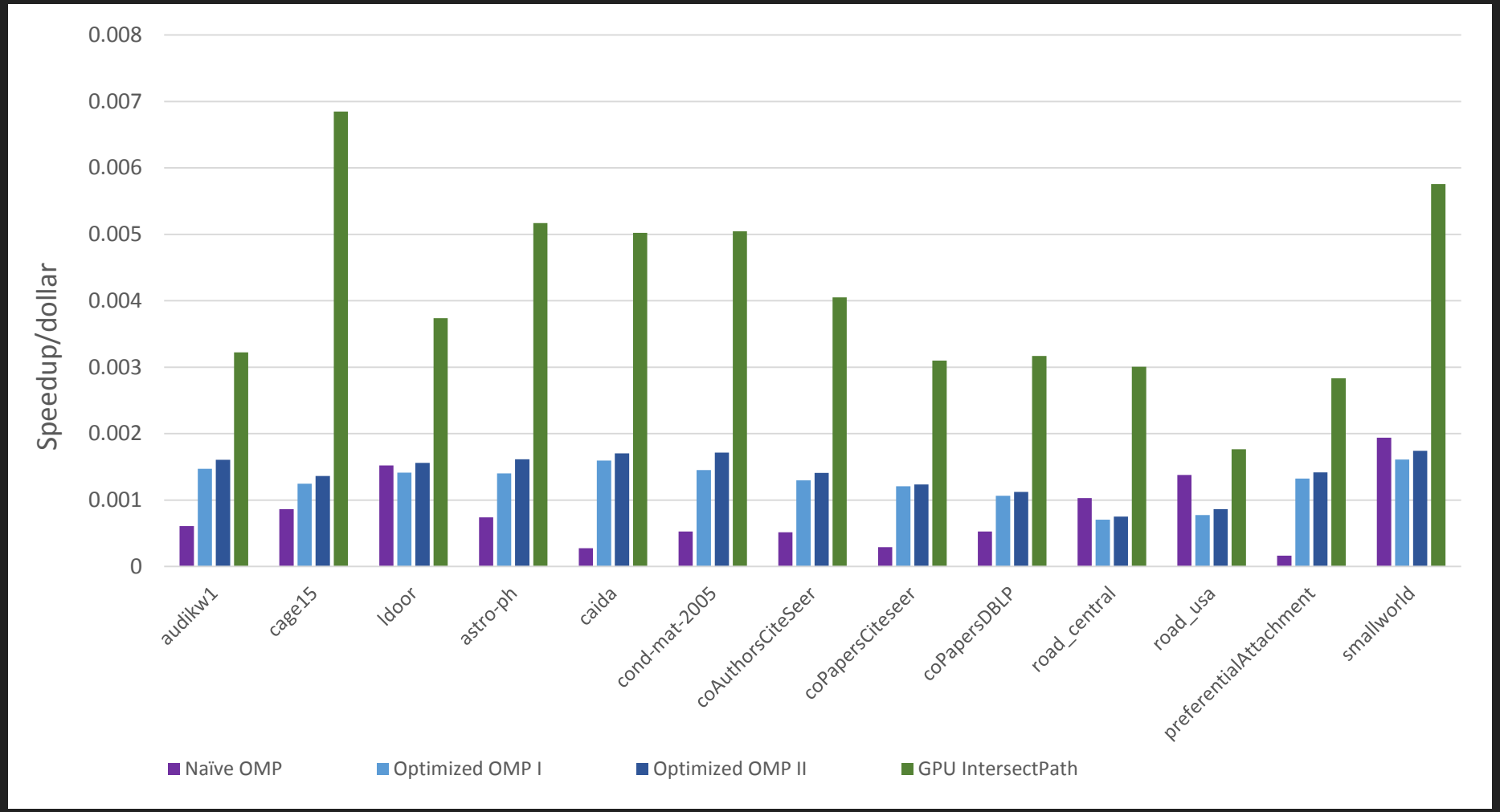


Sources: Facebook, Twitter

GPU Vs. CPU

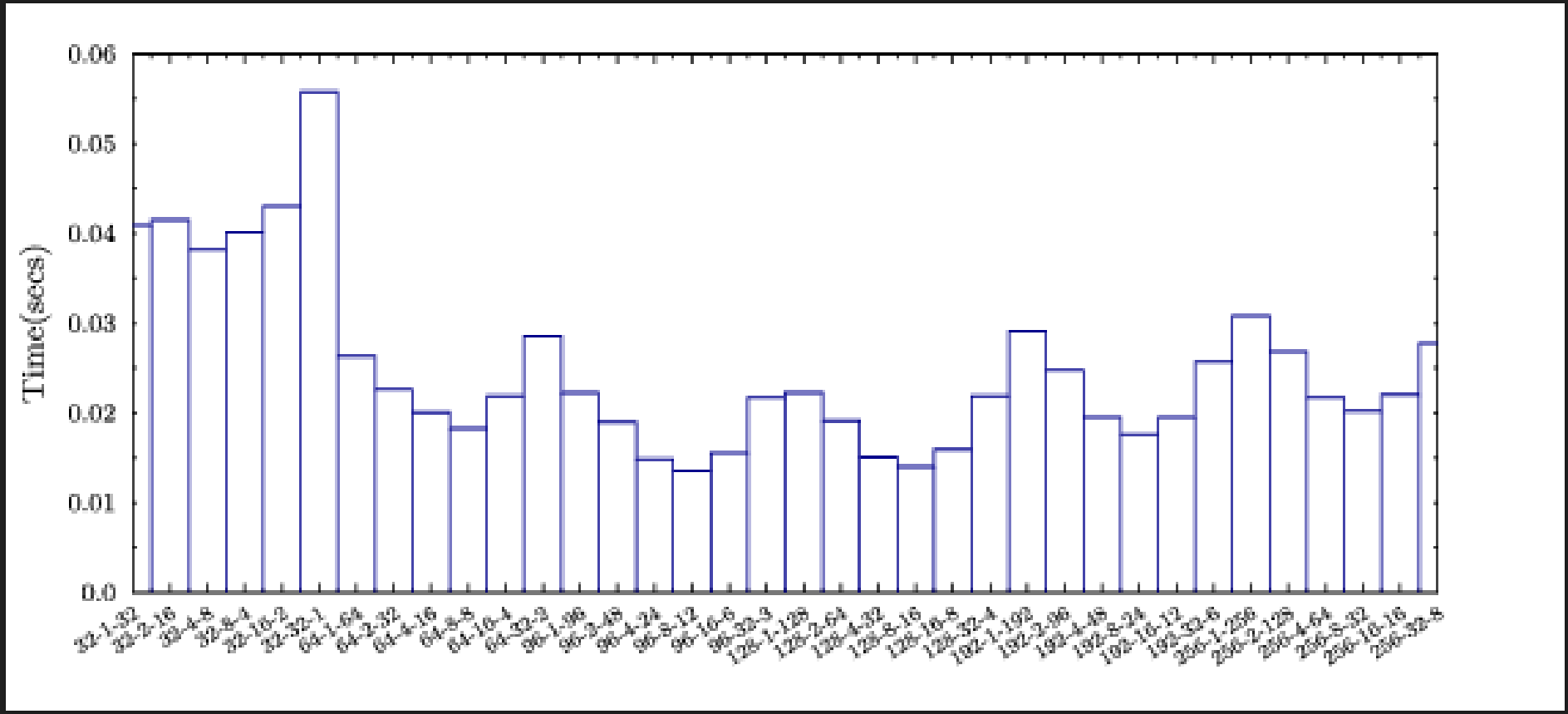

Speedup
40
35
30
25
20
15
10
5
0
audikw1
cage15
ldoor
astro-ph
caida
cond-mat-2005
coAuthorsCiteSeer
coPapersCiteseer
coPapersDBLP
road_central
road_usa
preferentialAttachment
smallworld
Naïve OMP
Optimized OMP I
Optimized OMP II
GPU IntersectPath
Sequential (Baseline)
Maximal for 40 cores

Speedup/Dollar



- Assuming
  - CPU : $20K
  - GPU : $5K

GPU – Parameter control



Growing thread block size