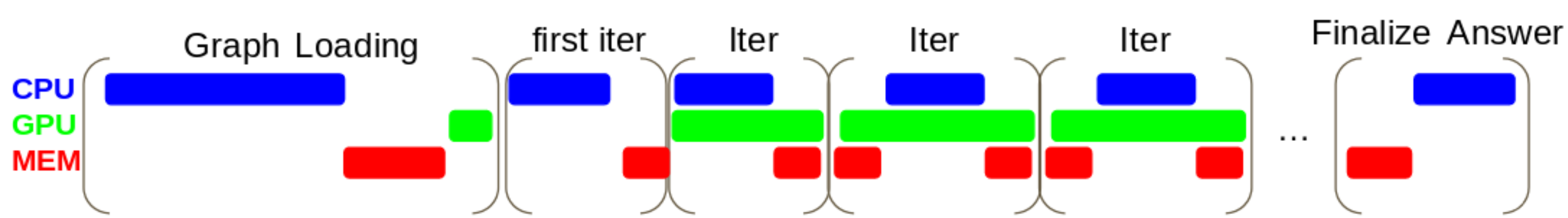
CPU/GPU Workload Harmony for BFS

How efficiently can we compute SSSP using BFS by using both the CPU & GPU?

- BFS: Breadth-First Search - each iteration moves one unit of distance further
- SSSP: Single Source Shortest Path - find distance from every vertex to some root
- 8-thread/4-core 3.20 GHz CPU (Intel i7-960)
- 2560-core/80-warp/20-SM 1.61 GHz GPU (Nvidia GTX 1080)

Current Implementation:

- GPU and CPU have independent vertex sets that they process in parallel
- Communicate each other's seen vertices between iterations to synchronize
- Goal: Maximize overlap b/w computation and communication



Implementation

Frontier representation:

- Full (Boolean vs. Bit Vectors)
- Sparse

Synchronization:

- After every iteration
- After multiple iterations

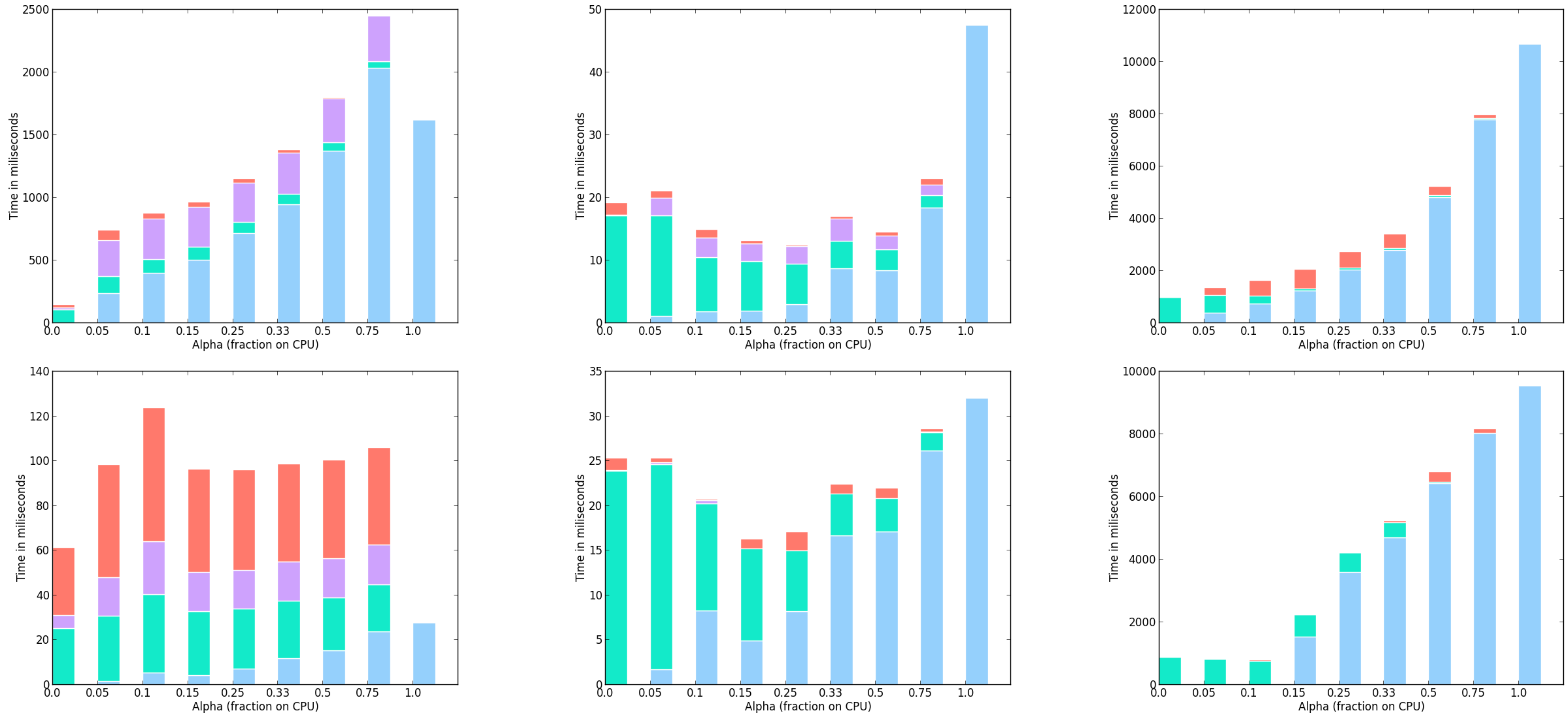
Partitioning:

- Static: Statically assign some fraction of the graph edges to the CPU and the rest to the GPU
  - Random
  - Degree-based: Sort the vertices based on degree and then partition between the two processors
- Dynamic
  - A work-queue of frontiers : CPU and GPU pick work dynamically
  - Mid and small sized graphs reside on both the CPU and GPU

Full vs. Sparse Frontiers

FULL (BOOLEAN)

SPARSE



grid1000x1000

~1m vertices ~4m edges

Avg Degree: 3.996

com-youtube_3m

~1m vertices ~3m edges

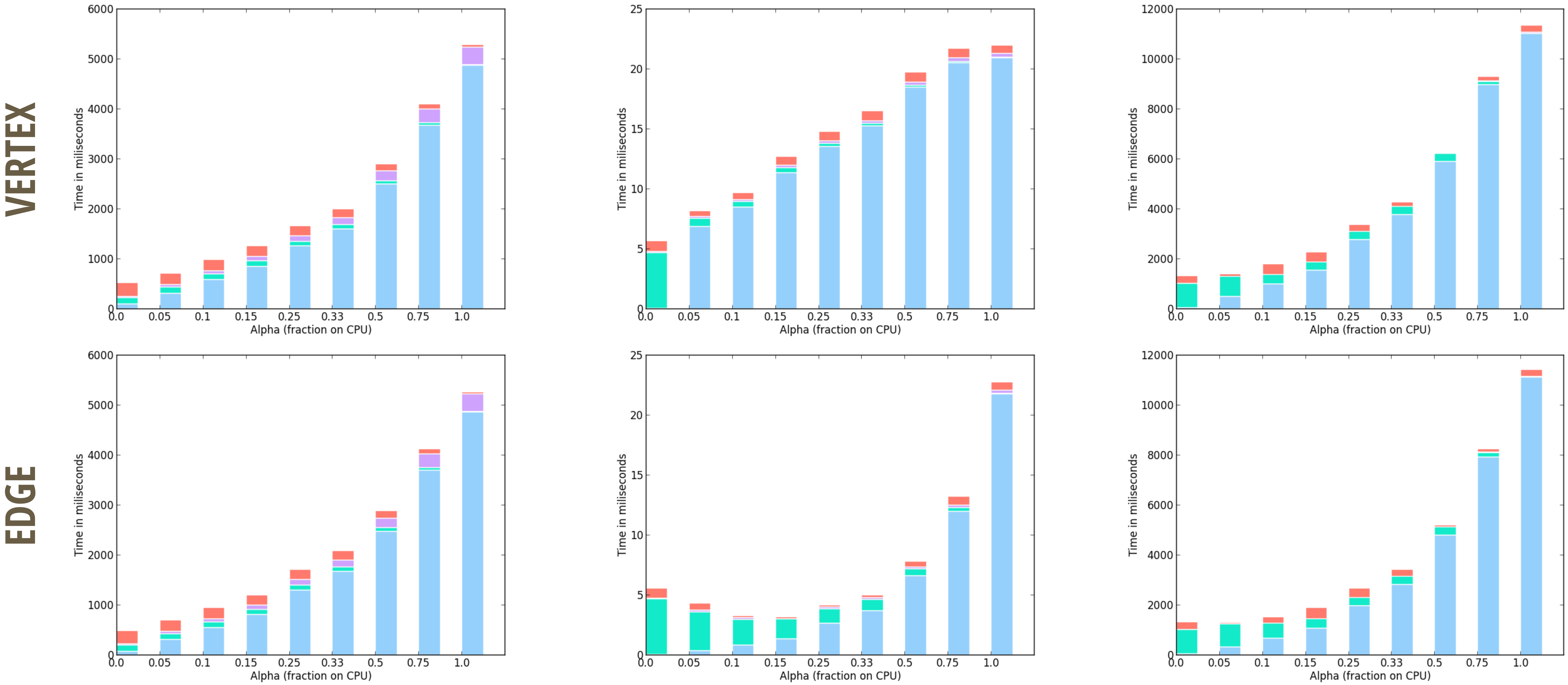
Avg Degree: 2.653

random_500m

50m vertices ~500m edges

Avg Degree: 10

Vertex vs. Edge Based Partitioning



grid1000x1000

~1m vertices ~4m edges

Avg Degree: 3.996

ego_twitter_2m

~80k vertices ~2m edges

Avg Degree: 30

random_500m

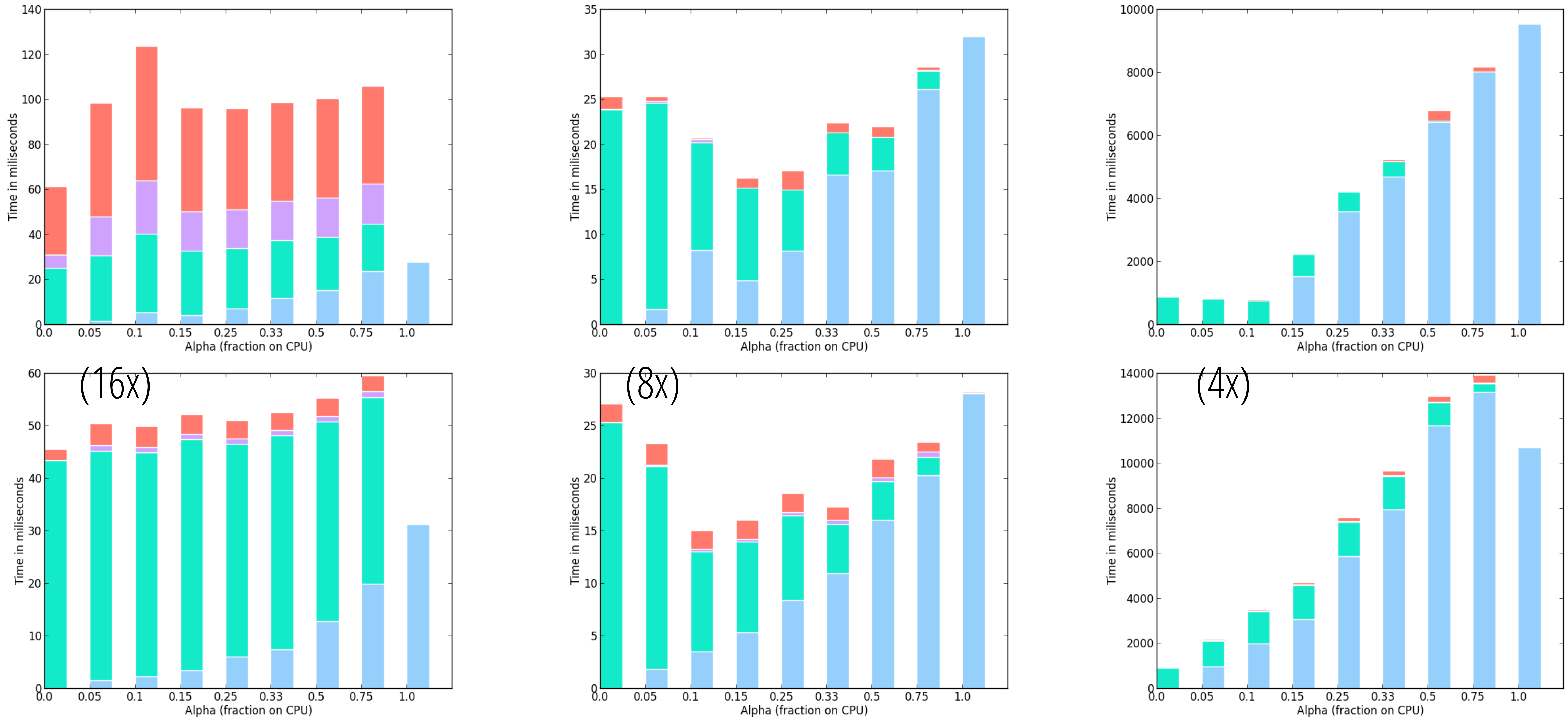
50m vertices ~500m edges

Avg Degree: 10

Single vs. Multiple Iterations b/w Synchronization

SINGLE

MULTIPLE



grid1000x1000

~1m vertices ~4m edges

Avg Degree: 3.996

com-youtube_3m

~1m vertices ~3m edges

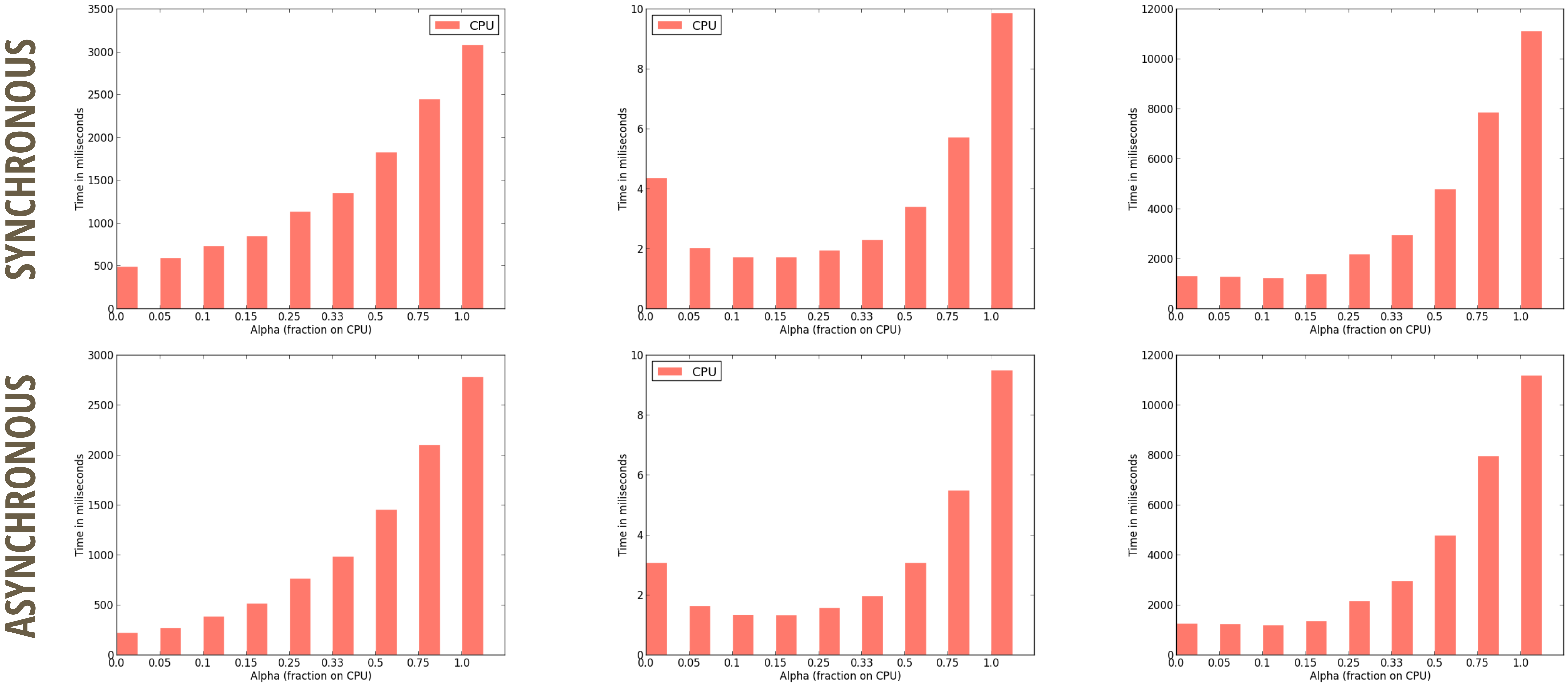
Avg Degree: 2.653

random_500m

50m vertices ~500m edges

Avg Degree: 10

Synchronous vs. Asynchronous Transfers



grid1000x1000
~1m vertices ~4m edges
Avg Degree: 3.996

soc-slashdot_900k
~80k vertices ~900k edges
Avg Degree: 11.54

random_500m
50m vertices ~500m edges
Avg Degree: 10

What're we doing next

- Standardize across implementations and optimize even further
- Merge ideas from different implementations to create new strategies/ approaches
- At runtime, pick the best strategy based on graph analysis and statistics

What we borrowed

- **Publications**
  - Efficient Large-Scale Graph Processing on Hybrid CPU and GPU Systems https://arxiv.org/pdf/1312.3018.pdf
  - HyGraph: Fast Graph Processing on Hybrid CPU-GPU Platforms by Dynamic Load-Balancing http://materials.dagstuhl.de/files/17/17431/17431.AnaLuciaVarbanescu1.Preprint.pdf

- **Graphs**
  - Stanford Large Network Dataset Collection
    https://snap.stanford.edu/data/

- **Starter Setup Code** (for graph importing)
  - CMU 15418 Spring 2017
    http://15418.courses.cs.cmu.edu/spring2017/article/7