SAND2020-10981 C

Rensselaer

Sandia National Laboratories

Distributed Graph Coloring on Multiple GPUs

Ian Bogle - RPI/Sandia National Laboratories

Erik Boman, Karen Devine, Siva Rajamanickam – Sandia National Laboratories

George Slota - RPI

We thank the Center of Computational Innovations at RPI for maintaining the equipment used in this research, including the AiMOS supercomputer supported by the National Science Foundation under Grant No. 1828083. This research was also supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Main contributions and results

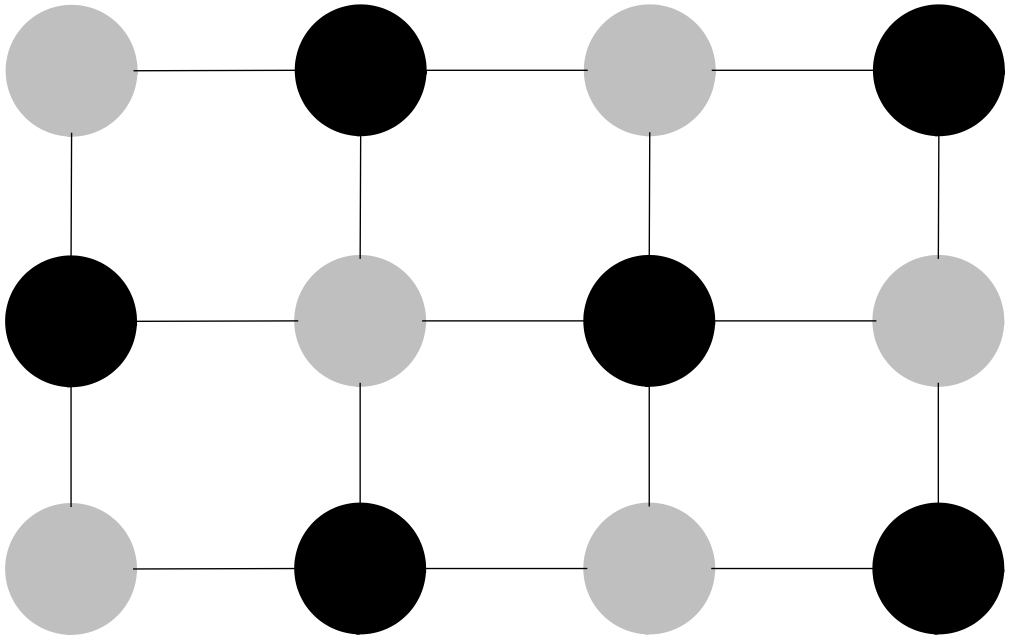
- We present the first distributed memory multi-GPU graph coloring implementation, to our knowledge
- Our distributed distance-1 coloring implementation sees up to a **28x speedup** on 128 GPUs over a single GPU, and **only a 7.5% increase** in colors on average
- Our distributed distance-2 coloring implementation also sees up to a **28x speedup** on 128 GPUs over a single GPU, and a **4.9% increase** in colors in the **worst** case
- Our approach is able to color a 12.8 Billion vertex mesh with **76.7 Billion** edges in **under half a second**
- We also present an approach that reduces the number of collective communications by increasing the cost of each communication.

Graph Coloring is Useful for Finding Concurrency

- Computations with specific data access patterns can be parallelized with colorings
- Compiler parallelization, register allocation
- Preprocessing Jacobian and Hessian matrix computations
- Also useful for finding short-circuits in circuit designs

Distance-1 Graph Coloring

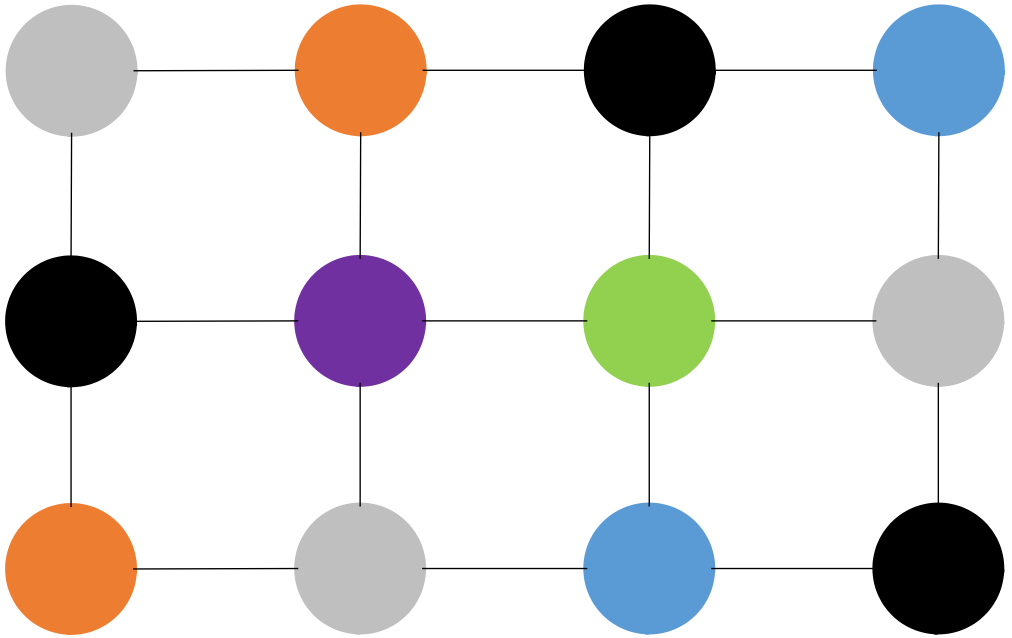
Each vertex ends up with a different color from its neighbors.



Minimizing the number of colors used in the coloring is known to be NP-Hard.

Distance-2 Graph Coloring

A valid distance-2 coloring assigns each vertex a color not used by any vertex at most two edges away



Minimizing the number of colors used in the coloring is known to be NP-Hard.

Related Work

- Coloring heuristics use few enough colors to be useful to applications
- Two main parallelization approaches
  - Jones & Plassmann (1993) uses independent sets to color vertices in parallel
  - “Speculate & Iterate” approaches use heuristics in parallel and fix conflicts
    - Gebremedhin and Manne (2000)
    - More popular approach recently
- Shared Memory & GPU Implementations
  - Catalyurek et al. (2012) and Rokos et al. (2015) present shared-memory implementations
  - Deveci et al. (2016) and Grosset et al. (2011) present coloring implementations on the GPU
- Distributed approaches
  - Bozdăg et al. (2008) adapt the “Speculate & Iterate” approach in distributed memory, subsequent works (2010) include a distance-2 implementation.
  - Sariyüce et al. (2012) presents a hybrid MPI+OpenMP implementation.

We Leverage KokkosKernels to Run on GPUs

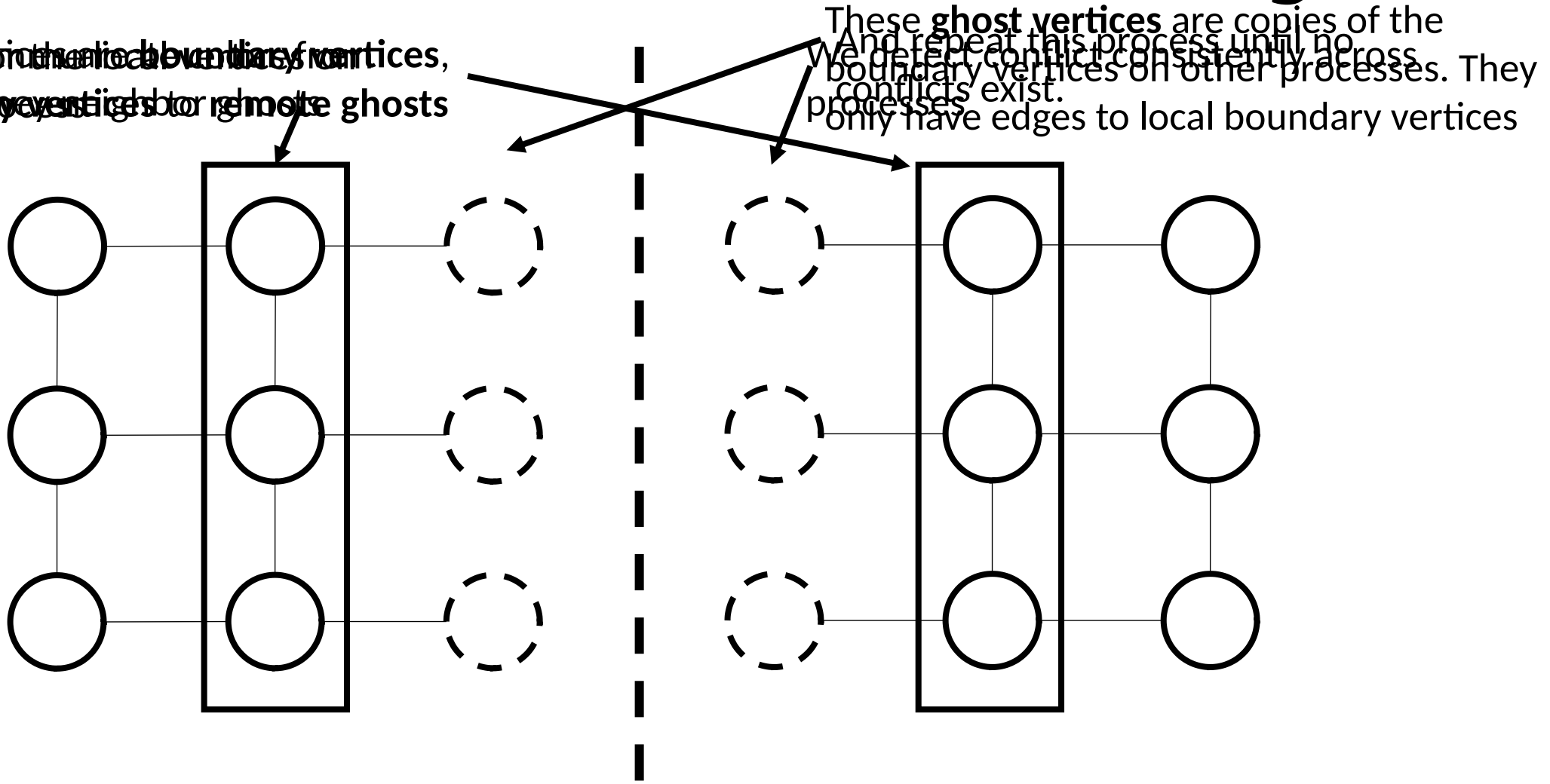
- KokkosKernels is a package that implements various linear algebra and graph operations, using the Kokkos shared-memory parallel programming model
  - https://github.com/kokkos/kokkos-kernels
- The distance-1 graph coloring algorithms we use are described by Deveci et al. (2016)
- The distance-2 graph coloring algorithm in KokkosKernels uses the same Net-Based approach described by Taç et al. (2017)
- We vary the distance-1 algorithms based on how skewed the inputs are.

Distributed “Speculate and Iterate” Coloring

- Based on Gebremedhin and Manne (2000)

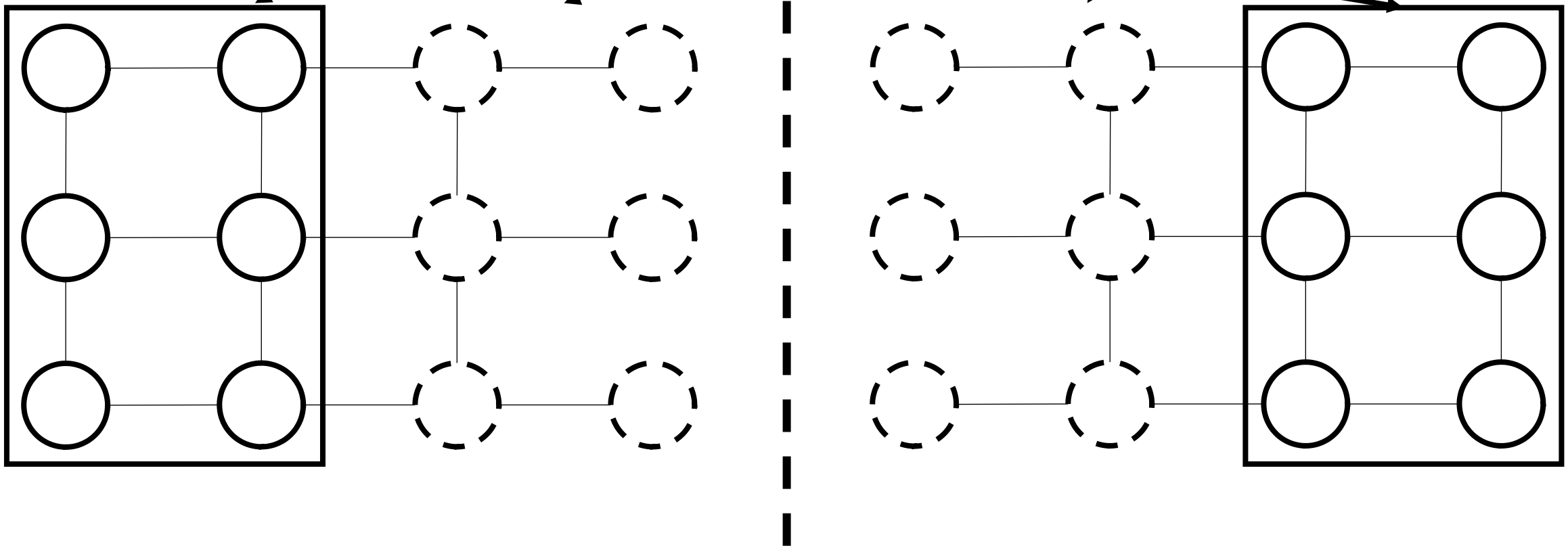
procedure PARALLEL-COLOR(Graph $G = (V, E)$)
    Color all local vertices
    Communicate colors of boundary vertices
    do
        Detect conflicts
        Recolor conflicting vertices
        Communicate updated boundary colors
    while Conflicts exist

Distributed Distance-1 Coloring



Distributed Distance-2 Coloring

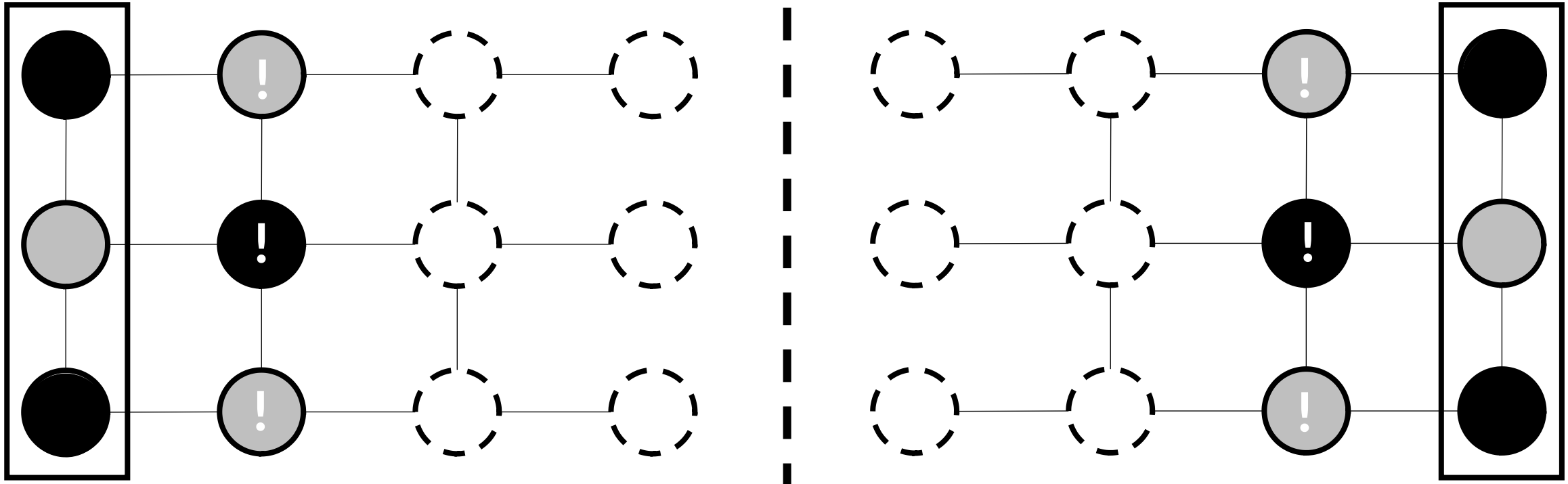
We add another layer of **ghost vertices** by copying the first ghost layer's neighbors. Second layer ghosts only have edges to the first ghost layer



Distance-1 with Two Ghost Layers (D1-2GL)

These ghosts will not conflict with any ghosts

Each process can resolve conflicts independently, without communicating (in this example)

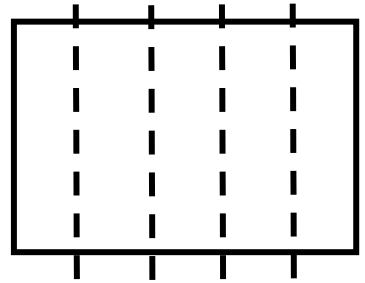


D1-2GL Aims To Reduce the Total Number of Collective Communications

- Second ghost layer vertices can get recolored, in general
- Local colorings need to make the same choices independently
- Each communication costs more
  - Ghosts can be on more than one process
  - Each ghost copy also copies its neighbors
  - Very expensive for dense inputs

Our Results Include Real and Synthetic Inputs

- Our real inputs come from a variety of domains
  - Including: PDEs, social networks, road networks
  - Range in size from 0.9 M vtx, 21 M edges to 30 M vtx, 3.3 B edges
  - Max degrees vary from 13 to 2.9 M
  - Use XtraPuLP graph partitioner developed by Slota et al. (2017)
- Our synthetic inputs are uniform hexahedral meshes
  - Vary only one dimension to keep communication costs constant
  - Partitioned in slabs
  - Range in size from 12.5 M vtx, 75 M edges to 12.8 B vtx, 76.7 B edges



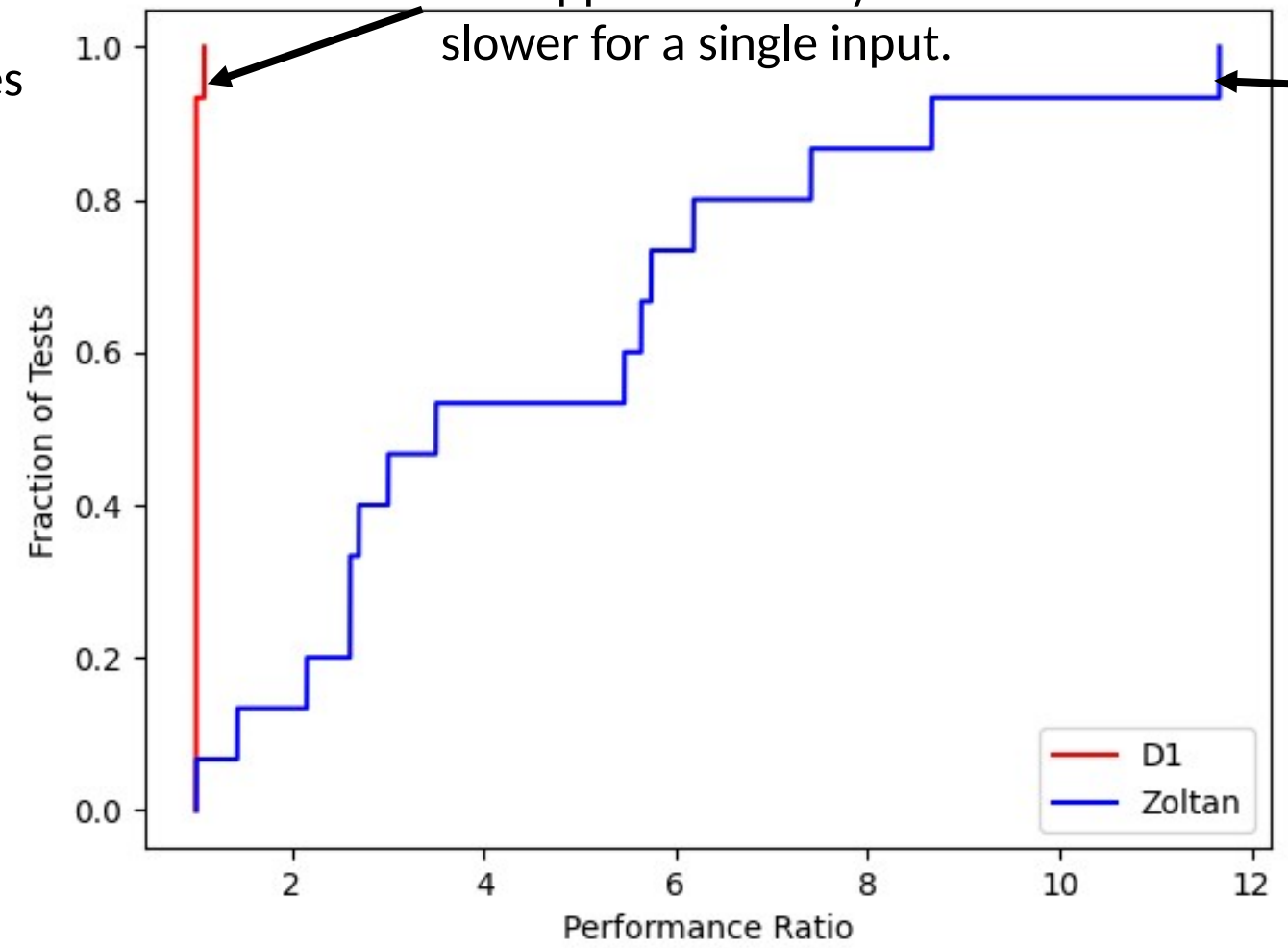
Experimental Setup

- We performed tests on AiMOS, housed at RPI.
  - 268 nodes with 2 IBM Power9 3.15GHz processors
    - 4 NVIDIA Tesla V100 GPUs with 16GB of memory connected with NVLink
  - Infiniband interconnect
  - Compiled with xlC 16.1.1, and Spectrum MPI with GPU-Direct disabled
- We run all experiments with 4 ranks per node
  - On GPU runs each rank gets an exclusive GPU
  - For performance profiles we use 128 ranks, or the largest run for which all approaches completed a run
- Zoltan is a package of the Trilinos Library containing distributed combinatorial algorithms, has MPI-only distance-1 and distance-2 implementations
  - http://cs.sandia.gov/zoltan
  - Run with 4 ranks per node, no shared-memory parallelism

Distance-1 Runtime Performance Profile

Performance ratio can be thought of as how many times slower one approach is than the other

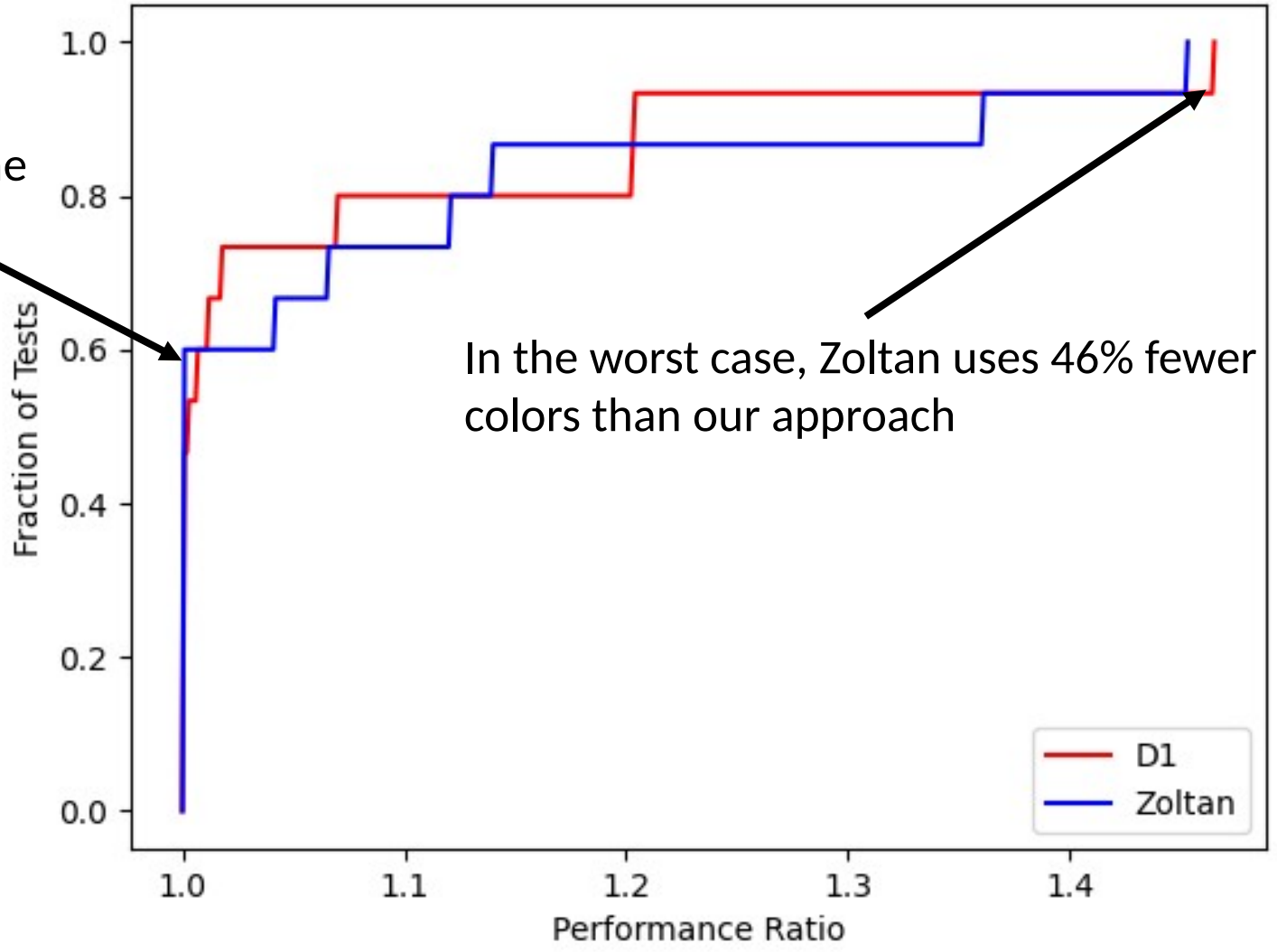
Our approach is only **8%** slower for a single input.



Our approach is around **12x** faster than Zoltan in the best case

Distance-1 Color Usage Performance Profile

Zoltan uses fewer colors in 60% of the inputs

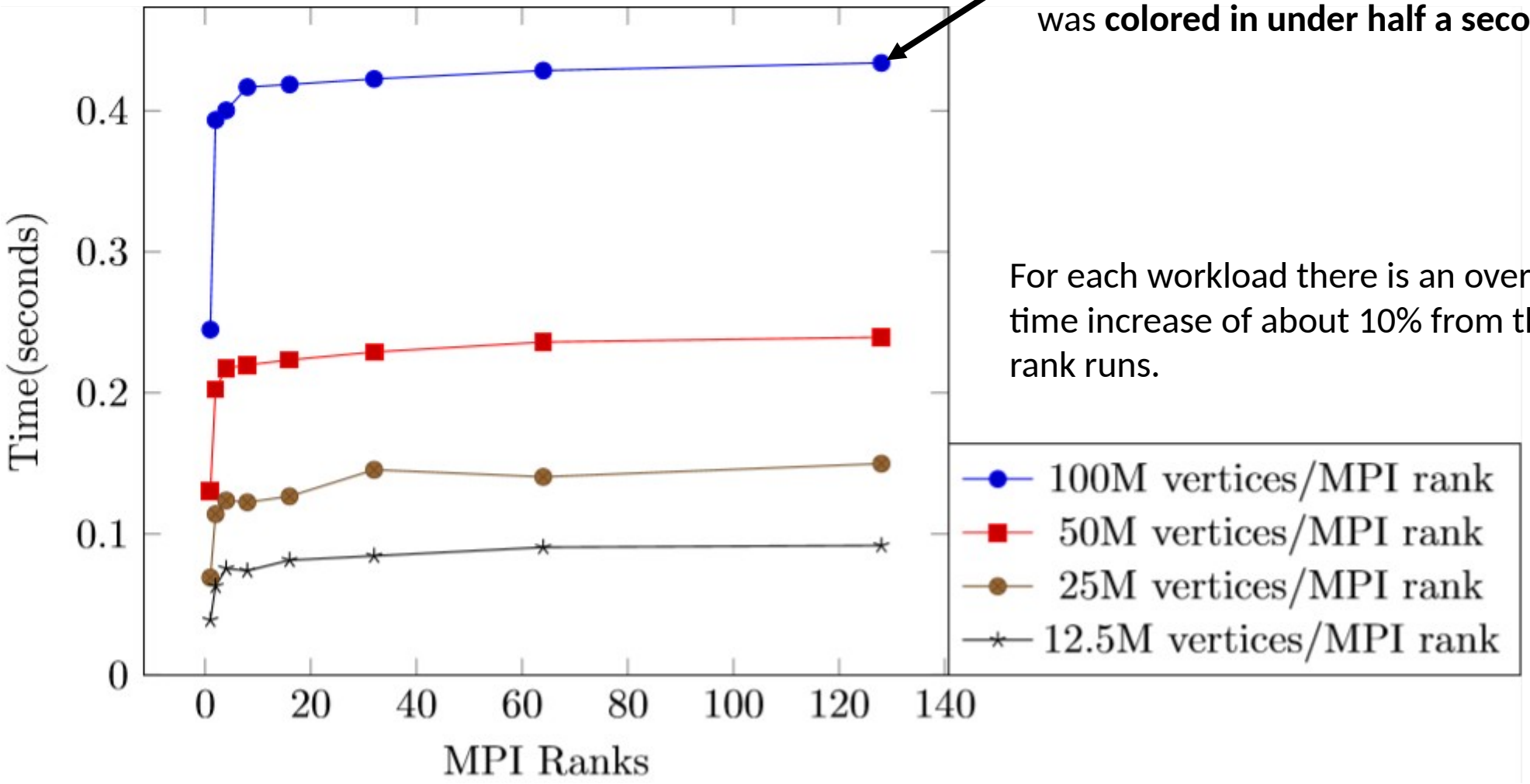


In the worst case, Zoltan uses 46% fewer colors than our approach

On average we use **6.8% more colors** than Zoltan

Distance-1 Weak Scaling

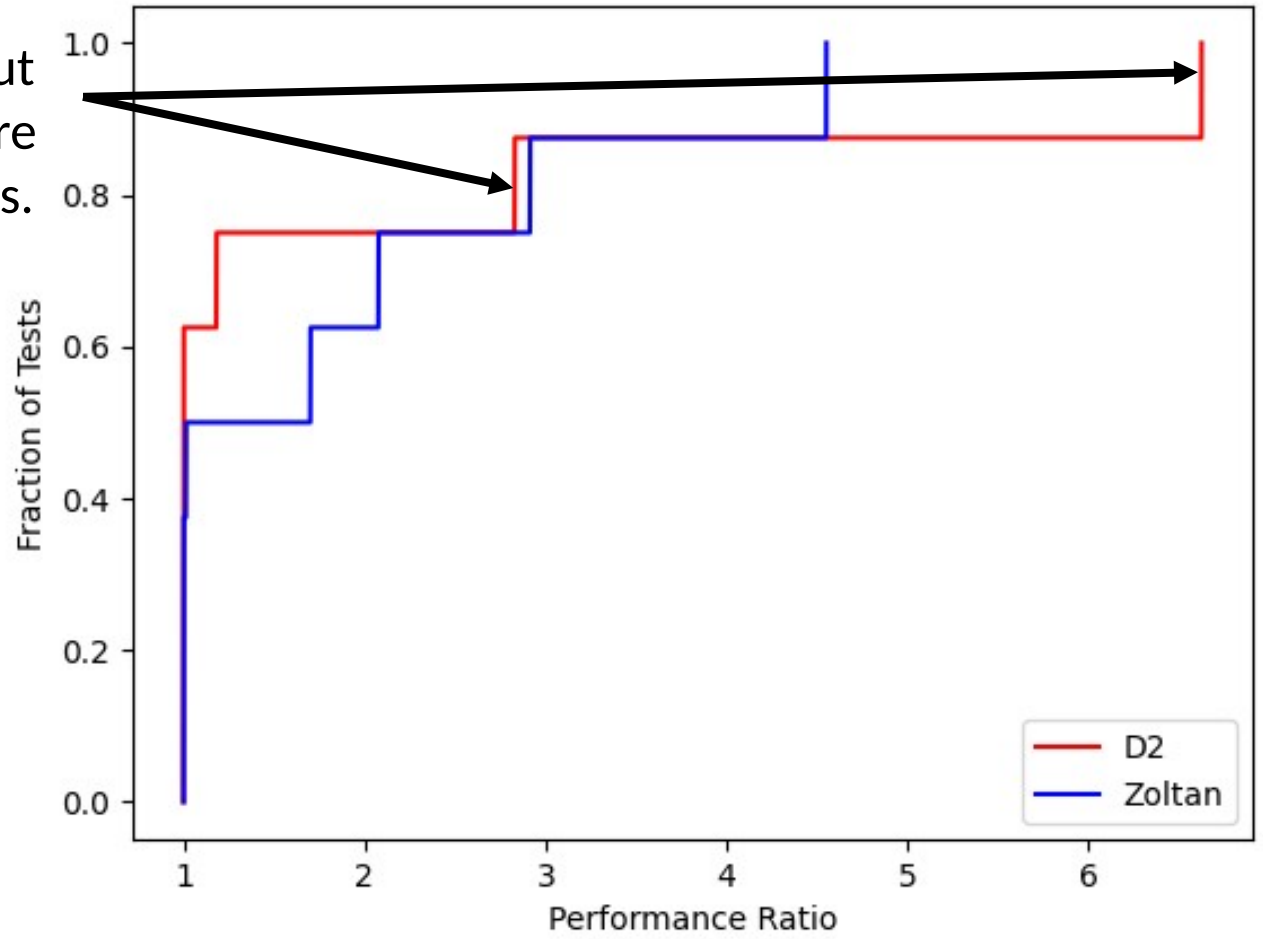
The largest input we ran has **12.8 Billion vertices** and **76.7 Billion edges**, which was **colored in under half a second**.



For each workload there is an overall time increase of about 10% from the 2 rank runs.

Distance-2 Runtime Performance Profile

We are competitive with Zoltan for all but two inputs, which are highly skewed inputs.

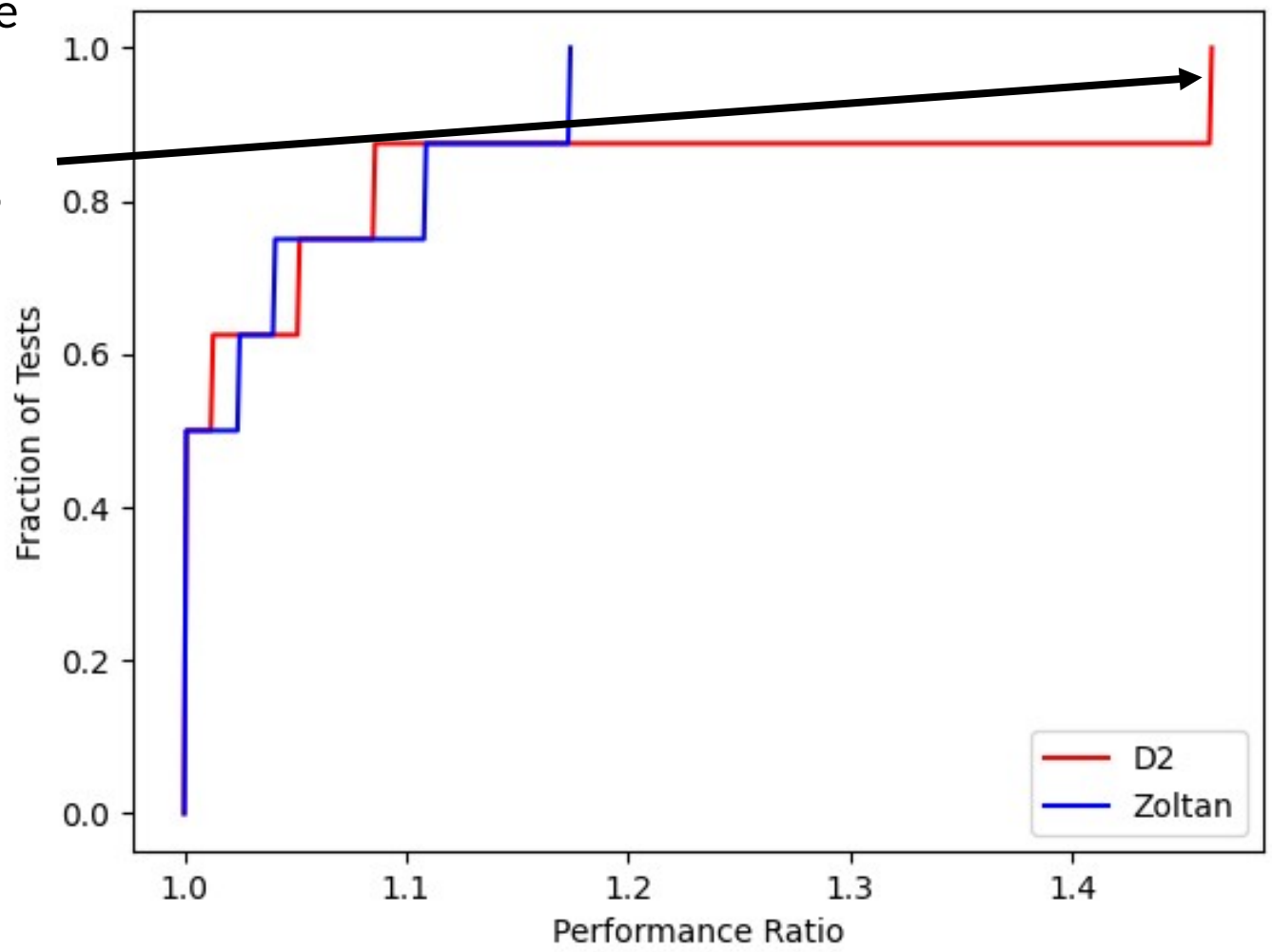


We use a smaller set of inputs for these profiles

Our approach sees at most a **4.5x speedup** relative to Zoltan

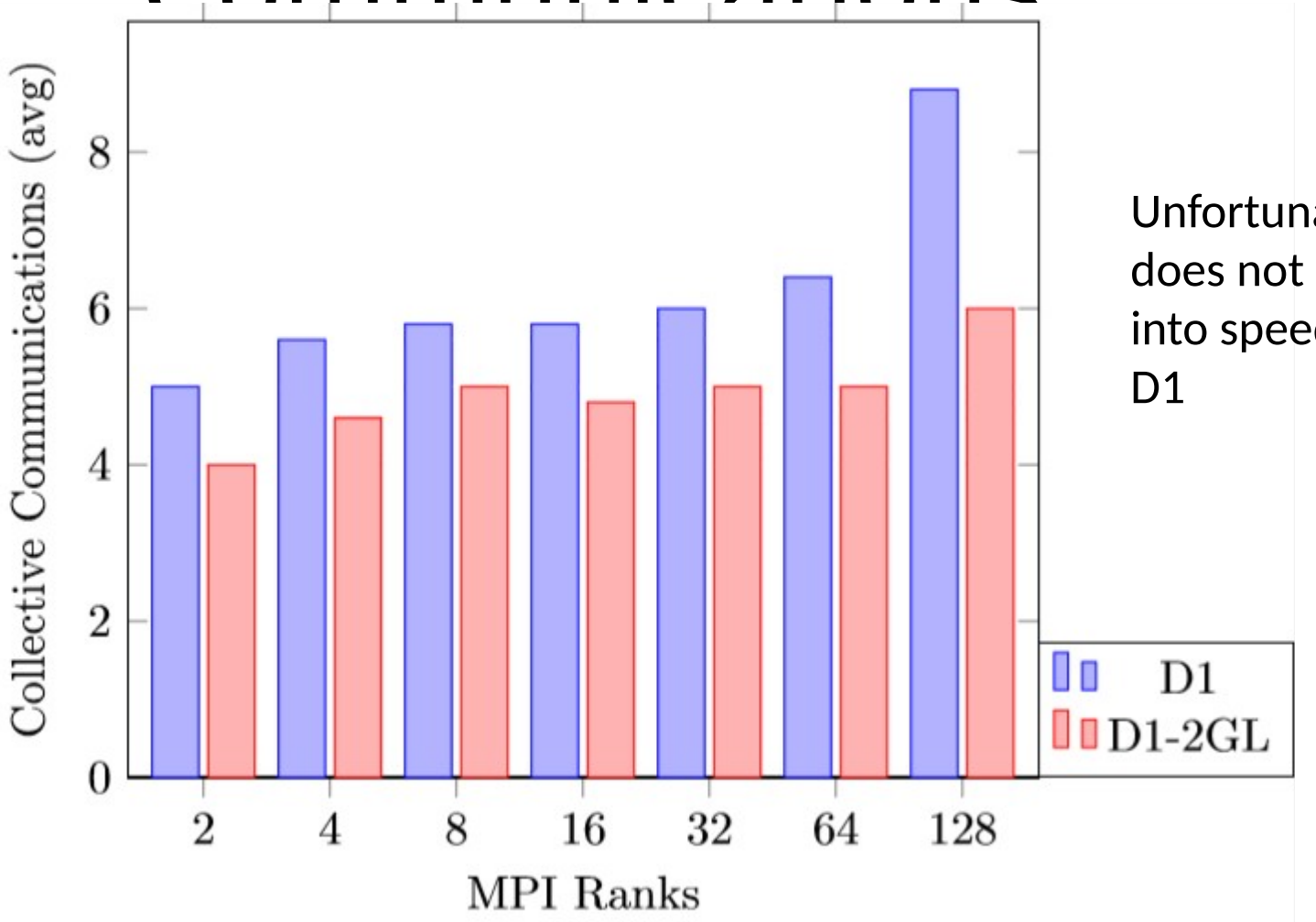
Distance-2 Color Usage Performance Profile

We are competitive with Zoltan on all but one input, where we use 46% more colors than Zoltan.



Our approach uses at most 10% more colors than Zoltan in all but one case

Two Ghost Layers Reduce the Number of Collective Communications



Unfortunately, this does not translate into speedup over D1

Future Work

- Optimizing distance-2 communication
- Extending our approach to solve different coloring problems
- Exploring optimizations to D1-2GL communication

Main contributions and results

- We present the first distributed memory multi-GPU graph coloring implementation, to our knowledge
- Our distributed distance-1 coloring implementation sees up to a **28x speedup** on 128 GPUs over a single GPU, and **only a 7.5% increase** in colors on average
- Our distributed distance-2 coloring implementation also sees up to a **28x speedup** on 128 GPUs over a single GPU, and a **4.9% increase** in colors in the **worst** case
- Our approach is able to color a 12.8 Billion vertex mesh with **76.7 Billion** edges in **under half a second**
- We also present an approach that reduces the number of collective communications by increasing the cost of each communication.

- Contact Me: boglei@rpi.edu