Performance Optimization on Massively Parallel Platforms with Heterogeneous Memory Architectures

Bo Wu
Colorado School of Mines

Motivation

- Massively parallel platforms require heterogeneous memory architectures for optimal performance
  - Nvidia GPUs have shared memory, device memory, and can directly access host memory
  - Intel Knights Landing processors have on-package high-bandwidth memory
- The different memory modules have their unique performance characteristics
- Programmers have to fine-tune the applications to match the massive parallelism with the memory heterogeneity

Challenges

- The different memory modules have quite different constraints, such as capacity, latency, throughput, etc.
- Real-world applications show dynamic behaviors and input sensitivity
- Co-running applications contend to use the low-latency or high-bandwidth memory modules

Goals

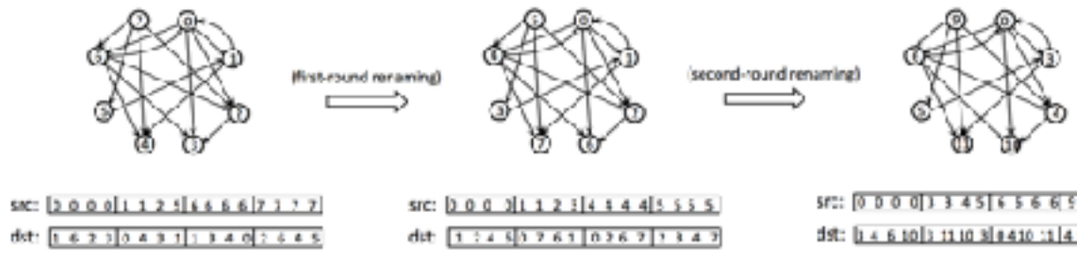
- Understand the interplay between program behaviors and different placement strategies
- Understand the trade-off between data migration and data placement
- Design a compiler and/or runtime optimization framework to automatically and transparently optimize performance
- Demonstrate the performance benefits on real-world complex applications in various scenarios

Case Study

Processing very large graphs on one single GPU

- The input graph is too large to fit in the device memory
- The graph applications typically need many super steps, resulting in redundant data transfers between the host memory and the device memory
- Existing approaches cannot efficiently use shared memory due to the irregularity

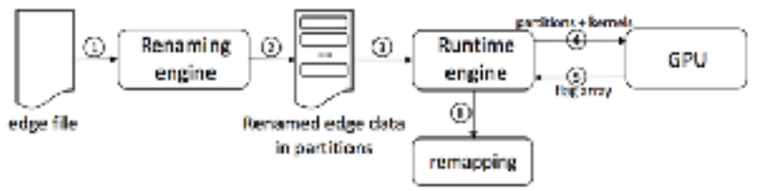
Our idea: vertex renaming



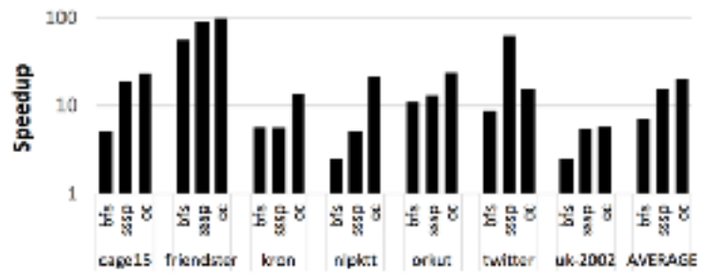
Enables efficient accesses to shared memory for source vertices

Enables quickly activating partitions to transfer in the next super step

The framework



Preliminary results



bwu@mines.edu