Exploring Hybrid Hardware and Data Placement Strategies for the Graph 500 Challenge

Experimental Platform

- **Software Platform**
  - **Tinker** is a graph processing framework that simplifies the task of implementing graph algorithms for single-node platforms.
  - **Snoopy** is a synchronous parallel (SOP) model and provides key functionalities to implementers:
    - User data representation
    - Ghost nodes and their synchronization
    - Needs to call back that implement algorithm specific (CPU and GPU) kernels.
    - Common optimizations (e.g., message aggregation, and overlapping computation with communication).
  - **Tinker** is open source.
  - netlab.ece.ubc.ca/wiki/index.php/Snoopy

- **Algorithms**
  - We use a standard Top-Down, level synchronized Breath First Search algorithm.
  - The CPU and GPU each have their own kernel, which is similar to the implementations proposed by Youg et al. (PACT 2011, PACT 2012)

Memory Layout of the Graph

- **Graphs** are represented in Compressed Sparse Row format, which is memory efficient.
  - The vertices are represented in an array with the values being the degree of the vertex.
  - The location of the vertices’ neighbors is the prefix sum of all previous vertices.
  - The vertex IDs are initially implicit as the index of the vertex.
  - Each undirected edge in the graph is represented as two directed edges.
  - We optionally sort the graph by each vertex’s degree, while holding onto the original ID.
  - We sort the graph back after executing any analysis on the graph.
  - This does not impose any change to the BFS algorithm.

The Configuration Space

- **Hardware Platform**
  - Two Intel Xeon E5-2670 processors (2.5 GHz, 10 cores, 32GB memory)
  - Two Nvidia Kepler K40 (745 MHz, 2880 cores, 1GB memory)
- **Software Platform**
  - We focus on graphs from Scale 27 to 30, this means up to 1 billion vertices and 16 billion undirected edges.
  - Fully sorting the graph improved performance for every type of configuration.
  - Memory configuration – Scale 30

Lessons Learned

- Changing the vertex ID to a descending order improves performance.
  - The hybrid configuration can take advantage of partitioning structure to achieve better performance on non-ideal graphs.
- Energy normalized performance is at least as good (if not better) than CPUs when adding GPUs.

What We Have Next

- We will use our hybrid framework to match hardware with workflows they can best fit to.
  - Previous work [Gharai, IPSYS’15] has shown that if we can execute a large amount of the work on the GPU, it is beneficial to place the high-degree vertices on the GPU.
  - Our work confirms that the results hold true as we increase graph size.
  - There is a sweet spot with ~30% of edges explored by CPUs (the rest by the GPUs).
  - The GPUs are able to process a high degree vertices' many neighbors in parallel.
- Two bars show Scale 27 and 30. The trend of these 3 sweet spots for our framework across the full order of magnitude.

The Impact of Sorting

- We can sort during the hybrid partitioning setup phase, at a significant performance benefit.
  - Fully sorting the graph improves performance for every type of configuration.
  - We explore the impact of approximate vertex sorting, by using a MB/MBs odd-even sort with variable precision.
  - Sorting using the VHT algorithm is highly effective.
  - High precision is necessary to achieve a good performance improvement.
  - Locality and caching play a huge role in this result.

Partitioning Across CPUs & GPUs

- **Energy and Normalized Performance**
  - The graph shows a spike analysis of the performance metric of Megareps/Watt, used in the Graph 500 Challenge.
  - We can see the results for adding GPUs is relatively close energy wise in comparison to just CPUs in the best case.
  - The time and effort spent using Mapped Memory to load the graph contributes to not seeing a boost in energy savings when using GPUs.

Lessons Learned

- Changing the vertex ID to a descending order improves performance.
  - The hybrid configuration can take advantage of partitioning structure to achieve better performance on non-ideal graphs.
- Energy normalized performance is at least as good (if not better) than CPUs when adding GPUs.

What We Have Next

- We will use our hybrid framework to match hardware with workflows they can best fit to.
  - Previous work [Gharai, IPSYS’15] has shown that if we can execute a large amount of the work on the GPU, it is beneficial to place the high-degree vertices on the GPU.
  - Our work confirms that the results hold true as we increase graph size.
  - There is a sweet spot with ~30% of edges explored by CPUs (the rest by the GPUs).
  - The GPUs are able to process a high degree vertices' many neighbors in parallel.
- Two bars show Scale 27 and 30. The trend of these 3 sweet spots for our framework across the full order of magnitude.

The Energy Case

- This graph shows a spike analysis of the performance metric of Megareps/Watt, used in the Graph 500 Challenge.
  - We can see the results for adding GPUs is relatively close energy wise in comparison to just CPUs in the best case.
  - The time and effort spent using Mapped Memory to load the graph contributes to not seeing a boost in energy savings when using GPUs.

Lessons Learned

- Changing the vertex ID to a descending order improves performance.
  - The hybrid configuration can take advantage of partitioning structure to achieve better performance on non-ideal graphs.
- Energy normalized performance is at least as good (if not better) than CPUs when adding GPUs.

What We Have Next

- We will use our hybrid framework to match hardware with workflows they can best fit to.
  - Previous work [Gharai, IPSYS’15] has shown that if we can execute a large amount of the work on the GPU, it is beneficial to place the high-degree vertices on the GPU.
  - Our work confirms that the results hold true as we increase graph size.
  - There is a sweet spot with ~30% of edges explored by CPUs (the rest by the GPUs).
  - The GPUs are able to process a high degree vertices' many neighbors in parallel.
- Two bars show Scale 27 and 30. The trend of these 3 sweet spots for our framework across the full order of magnitude.

The Energy Case

- This graph shows a spike analysis of the performance metric of Megareps/Watt, used in the Graph 500 Challenge.
  - We can see the results for adding GPUs is relatively close energy wise in comparison to just CPUs in the best case.
  - The time and effort spent using Mapped Memory to load the graph contributes to not seeing a boost in energy savings when using GPUs.