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

The Configuration Space

- The configuration space is large and significantly affects performance depending on many factors:
  - The partitioning of vertices across devices.
  - The order in memory of the graph arrays.
  - The memory access strategy for devices.
  - The degree of the vertices being explored.
- There are also some factors that are still not available to explore:
  - Access to distributed memory across devices.
  - The effect of the average node degree.
  - Does the sorting method help other algorithms?

Going Forward

- The configuration space is large and significantly affects performance depending on many factors:
  - The partitioning of vertices across devices.
  - The order in memory of the graph arrays.
  - The memory access strategy for devices.
  - The degree of the vertices being explored.

Partitioning Across CPUs and GPUs

- When partitioning, our goal is to match hardware with workloads they can best fit to.
- A previous paper from our lab gave some initial scenarios that can make considerable amount of work fit into the GPU. This is especially relevant on the CPU (See Code word: – small machine).
- Our work confirms that the results hold true as we put the graph size.
- There is a exact spot with 16% of edges being explored by CPUs (and the rest by the GPUs). The GPUs are able to process higher degree vertices’ memory footprint in parallel.
- These two graphs shows 27 and 30. The reaer for this exact spot continues to hold for both frameworks across this full-order of magnitude.

Using CUDA Mapped memory

- As we increase scale, the percentage of edges that can be offloaded to the GPU’s device memory become better.
- We used mapped memory to overcome this limitation with the expense of increased communication overhead.
- If we memory map both the vertex array and the edge array, the hybrid platform performs better than sorted graphs than if we just used the GPUs only.
- If we cannot at least keep the vertex array in the GPU memory and store edges, we can achieve almost a performance over the CPUs.
- Filling of the remaining GPU memory with properties of the edges can give us additional benefit.

The Energy Performance

- The GPU’s peak performance is achieved by mapping the vertex and edge arrays to the GPU’s device memory.
- We can see the results for adding GPUs is relatively close energy-wise in comparison to just CPUs in the best case.
- We believe the time and effort spent using Mapping Manager in the best BMX1200 is not using a boost in energy savings when using GPUs.

Scatter plot shows Tbps (Gbps in bytes) vs. Energy Usage Comparison (MTEPS/WATT)

Configuration Analysis

- Sorting graphs by vertex’s degree provides boost in all configurations, and it can be explicitly implemented with a memory map of the vertex array.
- Intuitive partitioning of the graph is important, using 2 CPUs and 2 GPUs and randomly partitioning is no better than just using the 2 CPU’s if we can sort.
- The hybrid configurations can take advantage of partitioning structure to achieve better performance on non-sorted graphs.

The Performance Energy

- The graph shows energy analysis of the performance metrics of MapTPE's per 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.
- We believe the time and effort spent using Mapping Manager in the best BMX1200 is not using a boost in energy savings when using GPUs.

Experimental Platform

- We use the Compressed Sparse Row format to represent graphs, which is
- The CPU and GPU each have their own kernels.
  - They are based off the implementations proposed by Ling et al.
  - They use the bulk-synchronous parallel (BSP) model and provides
- The GPU is able to process a high degree vertices’

Memory Layout of the Graph

- We use Compressed Sparse Row format to represent graphs, which is
- The vertex ID is initially implicit, in the index of the vertex. Each
- The vertex ID is implicit, in the index of the vertex. Each
- The vertex ID is implicit, in the index of the vertex. Each
- The vertex ID is implicit, in the index of the vertex. Each
- We can sort the CSR format by each vertices’ degree, which rearranges
- We can see the results for adding GPUs is relatively close

The Impact of Sorting

- We can sort during hybrid partitioning’s setup phase, at negligible cost.
- Fully sorting the graph improves every type of configuration.
- We composed the presence of vertex sorting, by using a MATLAB code with variable precision.
  - Sorting using the first half of the bit based sorted
  - Sorting using the first half of the bit based sorted
  - Sorting using the first half of the bit based sorted
  - Sorting using the first half of the bit based sorted
  - Sorting using the first half of the bit based sorted
  - Sorting using the first half of the bit based sorted

Lessons Learned

- Sorting graphs by vertex’s degree provides boost in all configurations, and it can be explicitly implemented with a memory map of the vertex array.
- Intuitive partitioning of the graph is important, using 2 CPUs and 2 GPUs and randomly partitioning is no better than just using the 2 CPU’s if we can sort.
- The hybrid configurations can take advantage of partitioning structure to achieve better performance on non-sorted graphs.

Data Placement

- Experimental Platform
  - The configuration space is large and significantly affects performance depending on many factors:
    - The partitioning of vertices across devices.
    - The order in memory of the graph arrays.
    - The memory access strategy for devices.
    - The degree of the vertices being explored.
- There are also some factors that are still not available to explore:
  - Access to distributed memory across devices.
  - The effect of the average node degree.
  - Does this sorting method help other algorithms?

- We use the Compressed Sparse Row format to represent graphs, which is
- The CPU and GPU each have their own kernels.
  - They are based off the implementations proposed by Ling et al.
  - They use the bulk-synchronous parallel (BSP) model and provides
- The GPU is able to process a high degree vertices’

- Memory Layout of the Graph
  - We use Compressed Sparse Row format to represent graphs, which is
  - The vertex ID is initially implicit, in the index of the vertex. Each
  - The vertex ID is implicit, in the index of the vertex. Each
  - The vertex ID is implicit, in the index of the vertex. Each
  - We can sort the CSR format by each vertices’ degree, which rearranges
  - We can see the results for adding GPUs is relatively close

- The Impact of Sorting
  - We can sort during hybrid partitioning’s setup phase, at negligible cost.
  - Fully sorting the graph improves every type of configuration.
  - We composed the presence of vertex sorting, by using a MATLAB code with variable precision.
    - Sorting using the first half of the bit based sorted
    - Sorting using the first half of the bit based sorted
    - Sorting using the first half of the bit based sorted
    - Sorting using the first half of the bit based sorted
    - Sorting using the first half of the bit based sorted
    - Sorting using the first half of the bit based sorted

- Lessons Learned
  - Sorting graphs by vertex’s degree provides boost in all configurations, and it can be explicitly implemented with a memory map of the vertex array.
  - Intuitive partitioning of the graph is important, using 2 CPUs and 2 GPUs and randomly partitioning is no better than just using the 2 CPU’s if we can sort.
  - The hybrid configurations can take advantage of partitioning structure to achieve better performance on non-sorted graphs.

Scott Salливан, Daniel Borges, Abdullah Gharabeh, Mati Ripeanu
sallivans.ece.ubc.ca
Electrical and Computer Engineering
The University of British Columbia