Abstract

Analysis of networks is quite interesting, because they can be interpreted for several purposes. Various features require different metrics to measure and interpret them. Measuring the relative importance of each vertex in a network is one of the most fundamental building blocks in network analysis. Betweenness Centrality (BC) is one such metric that plays a key role in many real-world applications. BC is an important graph analytics application for large-scale graphs. However it is one of the most computationally intensive kernels to execute, and measuring centrality in billion-scale graphs is quite challenging.

While there are several existing efforts towards parallelizing BC algorithms on multi-core CPUs and many-core GPUs, in this work, we propose a novel fine-grained CPU-GPU hybrid algorithm that partitions a graph into two partitions, one each for CPU and GPU. Our method performs BC computations for the graph on both the CPU and GPU resources simultaneously, resulting in a very small number of CPU-GPU synchronizations, hence taking less time for communications. The BC algorithm consists of two phases, the forward phase and the backward phase. In the forward phase, we initially find the paths that are needed by either partitions, after which each partition is executed on each processor in an asynchronous manner. We initially compute border matrices for each partition which stores the relative distances between each pair of border vertex in a partition. The matrices are used in the forward phase calculations of all the sources. In this way, our hybrid BC algorithm leverages the multi-source property inherent in the BC problem. We present proof of correctness and the bounds for the number of iterations for each source. We also perform a novel hybrid and asynchronous backward phase, in which each partition communicates with the other only when there is a path that crosses the partition, hence it performs minimal CPU-GPU synchronizations.

We use a variety of implementations for our work, like node-based and edge-based parallelism, which includes data-driven and topology based techniques. In the implementation we show that our method also works using variable partitioning technique. The technique partitions the graph into unequal parts accounting for the processing power of each processor. Our implementations achieve almost equal percentage of utilization on both the processors due to

the technique. For large scale graphs, the size of the border matrix also becomes large, hence to accommodate the matrix we present various techniques. The techniques use the properties inherent in the shortest path problem for reduction. We mention the drawbacks of performing shortest path computations on a large scale and also provide various solutions to it.

Evaluations using a large number of graphs with different characteristics show that our hybrid approach without variable partitioning and border matrix reduction gives 67% improvement in performance, and 64-98.5% less CPU-GPU communications than the state of art hybrid algorithm based on the popular Bulk Synchronous Paradigm (BSP) approach implemented in TOTEM. This shows our algorithm's strength which reduces the need for larger synchronizations. Implementing variable partitioning, border matrix reduction and backward phase optimizations on our hybrid algorithm provides upto 10x speedup. We compare our optimized implementation, with CPU and GPU standalone codes based on our forward phase and backward phase kernels, and show around 2-8x speedup over the CPU-only code and can accommodate large graphs that cannot be accommodated in the GPU-only code. We also show that our method‘s performance is competitive to the state of art multi-core CPU and performs 40-52% better than GPU implementations, on large graphs. We show the drawbacks of CPU and GPU only implementations and try to motivate the reader about the challenges that graph algorithms face in large scale computing, suggesting that a hybrid or distributed way of approaching the problem is a better way of overcoming the hurdles.