FINGERS: Exploiting Fine-Grained Parallelism in Graph Mining Accelerators

Spectredown (**# 4**)
Prajeeth.S
(190050117)

I. Summary

Graph Mining Algorithms have high time complexity which leads to a need for hardware accelerators. The current state-of-the-art design uses coarse-grained(Tree-level) parallelism, which has issues of hardware under-utilization and inefficient resource provision. The authors first identify fine-level parallelism, at the branch, set and the segment level and then propose FINGERS, which utilizes this to achieve significant performance improvements. On average, FINGERS gives a 2.8x speedup over the state-of-the-art design(FlexMiner).

II. Details

- Graph mining frameworks fall into two broad categories - pattern-oblivious(which searches through the whole space without using the structure of the pattern) and pattern-aware(which exploits the structure of the pattern to get a more efficient algorithm and used widely today).
- The Processing elements(PEs) are underutilized due to long memory stalls, suffer from inefficient resource provision and load imbalance in real-world graphs, because every DFS search tree is assigned to single PE, rather than parallelizing it.
- The authors 3 levels of fine-grained parallelism - branch level(which provides a generalization over the tree level by parallelizing the lower tree levels along with the choices for the root. This increases the number of available tasks to scheule in a single PE, but may lead to cache contention), set level(which parallelizes the set operations at every level and allows for data reuse) and segment level(which exploits the difference in the set sizes during computations, and hence dividing it into smaller segments, but requires careful load balancing design). FINGERS efficiently exploits these features.
- FINGERS contain multiple PEs(which inturn consists of multiple Intersect units(IU), that performs the set operations), connected through a Network-on-Chip(NoC). Each PE utilizes fine-level parallelism to speed up processing. The set and segment level parallelism are within a task, while the branch level parallelism affects the number of tasks available to a PE, where a ”task” refers to the process of extending a new vertex to the current embedding. These levels of parallelism are controlled by a load table, which determines the number of segments that can be run by a single IU.

III. Strengths

- FINGERS gives a significant performance of 2.8x speedup on average compared to FlexMiner.
- The usage of shared cache is more efficient(especially for smaller sized shared caches) because there are lesser PEs competing for capacity compared to FlexMiner.

IV. Weaknesses

- Although there is a significant performance improvement, it comes at a cost of twice the area compared to the FlexMiner chip.
- For graphs of large size and low degree, the memory access latency is starkly observed, and hence provides only a limited speedup.

V. Extensions

- The branch level parallelism has not been exploited well, and can be used to reduce the memory access latency for large graphs.
- An other alternative to overcome the same issue can be to combine processing-in-memory technology with FINGERS