Summary: Dynamic Thread Mapping for High-Performance, Power-Efficient Heterogeneous Many-core Systems
Guangshuo Liu, Jinpyo Park, Diana Marculescu

I. OVERVIEW

This paper investigates about the problem of dynamic thread mapping in heterogeneous many-core systems via an efficient algorithm that maximizes performance under power constraints. The approach is to formulate the mapping problem as a 0-1 integer linear program (ILP), given any numbers of threads, cores and type of cores. An iterative $O(n^2/m)$ heuristic-based algorithm for solving the 0-1 ILP thread mapping is proposed, thereby providing, a novel scalable approach for effective thread mapping for maximizing throughput on many-core heterogeneous systems.

The paper considers multi-threaded workloads and assumes that each core runs at most one thread at a time thereby supporting single threaded execution, without simultaneous multithreading (SMT). Simply, the total number of threads is assumed to be $n$, identical to the total number of cores. In the case that there are less threads than cores, dummy threads with zero throughput and power are considered mapped to idle cores. The threads are spawned in the beginning of the execution and that mapping decisions are taken at certain time intervals or control epochs, considering the number of threads is fixed throughout the duration of a control epoch.

With the increase in problem size, the existing ILP solvers prove to be inefficient in solving a 0-1 ILP problem, which are generally considered NP-complete, as the exponential increase in runtime can exceed the control epochs durations typically in range of milliseconds. Additionally, as the power constraint becomes tighter, the computation can take orders of magnitude times longer than a relatively unconstrained problem of the same size. The reason behind this inefficiency is that typical ILP algorithms are essentially based on branch-and-bound approaches that may iteratively search for the global optimum which may result in exceedingly high time complexities. Therefore, using ILP solvers is impractical for online dynamic mapping for heterogeneous many-core systems.

The proposed heuristic contains two phases, namely: maximization and swapping. The basic idea is to first aggressively assign threads such that highest possible throughput is achieved. Then, for adjacent core types, threads are swapped to reduce the total power. The paper describes the approach as maximization-then-swapping (MTS) heuristic. Note that the ordering here refers to the total ordering of core types in nominal power consumption. It does not have any relation to the topological placement of cores. Finally, if there is no possible swap, we consider the case to be infeasible. It indicates the power budget is too constrained to fit all the tasks using the heuristic.

In this paper, to validate the prediction model, workloads mixes of multithreaded benchmarks were randomly selected from PARSEC and SPLASH-2 benchmark suites. The data from offline profiling of PARSEC benchmarks was utilized and a binning approach was implemented to categorize unknown threads as their nearest neighbor in PARSEC benchmarks. The distance between an unknown thread and a profiled benchmark is measured as the difference in throughput on the core type that the unknown thread is running on.

II. CONTRIBUTION

Improvement over ILP solver

In the paper, the algorithm achieves runtime improvement of more than two orders of magnitude, compared with an efficient commercial ILP solver, while losing less than 0.6% total throughput on average. Up to 16% performance improvement is demonstrated under ISO-power constraints.

Scalability

The paper demonstrates that the heuristic scales to hundred-core systems with runtime overhead less than 1 ms, so that it can be brought online for large-scale thread mapping with relatively fine-grained control epochs.

Migration Cost

In the paper, the heuristic is validated to be history-aware for minimizing the migration cost in dynamic thread mapping. A tradeoff between throughput and migration cost is demonstrated.

III. LIMITATION

Load Balancing, No Simultaneous Multi threading

The paper confines to single thread per core which limits task migration between cores, which implies that sharing the threads is not considered, resulting in some tasks may miss deadlines. Load balancing is an aspect that needs to be addressed.

Fairness

The paper does not consider each process’s priority and workloads, but assigns the process to the cores depending on their types, which can result in CPU utilization loss, also effecting the QoS of such real time applications.