The era of manycore computing will bring new fundamental challenges in how we build, manage, and program on the computing system and its hardware. The techniques that were designed for single core processors will have to be dramatically changed to support the coming wave of extreme-scale general purpose parallel computing with thousands of cores on a single processor (see figure to the right). Today’s programming languages (e.g. C/C++, Java) are unlikely to scale to manycore levels given the level of expertise needed to parallelize applications; furthermore, their imperative nature makes them difficult to parallelize automatically. One approach to address such concurrency problem of future manycore system is to look at drastic alternatives, such as many-task computing (MTC), which has been applied to Grids and Supercomputers. Many MTC applications are structured as graphs of discrete tasks, with explicit input and output dependencies forming the directed graph edges. Given such kind of graph by which the traffic characteristic among the tasks is known a priori, how are the tasks scheduled to the many cores while achieving the best utilization and makespan? To address this issue, we designed both a static and dynamic scheduler, scalable to 1K-cores. The scheduling strategies are evaluated by using a cycle accurate NoC simulator NIRGAM. The simulation study shows that the proposed scheduling strategy result in 85% shorter makespan and 90% higher utilization in comparison to random mapping. In addition, our heuristic can tolerate the variance of the tasks’ execution times at runtime and deliver improved makespan and utilization.

Dynamic Scheduling Algorithm

This work addresses techniques for dynamically assigning the task to the processors to maximize the overall execution time and utilization rate at runtime. These include how to design an static and dynamic scheduler, as well as their execution overhead. The task scheduling problem for more than three processors is well known to be NP-complete. The methods aiming to find the optimal solution for the given objective are limited by the amount of time and memory needed since they grow as exponential function of the problem order. Because of the intractable nature of the scheduling problem, it is desirable to develop heuristic algorithms which could provide suboptimal solution within reasonable amount of computation time. Therefore, given an task graph with n nodes and manycore system with m cores, we introduce an heuristic with time complexity O(nm) to get an approximate optimal scheduling result on a 1K manycore system, which features 2D mesh topology due to their structural regularity and suitability for VLSI implementation. However, the heuristic could be easily applied onto other topologies (e.g., butterfly or fat tree). We evaluate the performance by measuring utilization and makespan, generated by the algorithm we proposed as well as an ad-hoc random algorithm. We use Task Graphs for Free (TGFF) benchmark to generate DAG with different number of task nodes and a cycle accurate NoC simulator NIRGAM to evaluate the performance result. We have developed a plug-in to the NIRGAM simulator for attaching the application to the NoC simulator, since the initial simulator only supports a uniform traffic pattern on the manycore topology.

Contributions

(1) The design and implementation of a heuristic based scheduler for DAGs, scalable to 1K-core processors and 16K node DAGs.
(2) The proposed scheduling heuristic can maximize utilization and reduce makespan, especially for data intensive applications.
(3) Proposed heuristic can tolerate variance or inaccuracies of task execution times at runtime.