Towards An Energy-Efficient Scheduler in the Codelet Model

Chen Chen[†], Yao Wu[†], Joshua Suetterlein[†], Long Zheng[‡§] and Guang R. Gao[†]
[†] Department of Electrical and Computer Engineering, University of Delaware, Newark, DE 19716, USA
[‡] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
[§] Department of Computer Science and Engineering, University of Aizu, Aizu-wakamatsu, 965-8580, Japan
Email: chenchen@capsl.udel.edu

***Abstract*—This paper uses the IBM Cyclops-64 many-core architecture as a case study to show how to exploit locality and save energy in the fine-grain dataflow-inspired codelet execution model. State-of-the-art codelet scheduling focuses on dynamic workload balance of codelets (similar to tasks). While this approach may achieve reasonable performance since computation resources are fully utilized, it may not attain optimal energy savings. In this paper, we propose a novel codelet scheduling algorithm that targets on energy efficiency. Our algorithm leverages static information regarding locality among codelets to achieve better performance and energy efficiency. By using local buffers to pass data produced in one codelet to another, both global memory accesses and energy consumption can be greatly reduced.**

I. Introduction

To continue to reach new levels of performance, HPC systems are growing extremely large and cumbersome. Current means for effectively utilizing these systems are quickly becoming antiquated. For this reason some are seeking alternate execution models parting from the unsatisfying MPI and OpenMP models which have dominated today's parallel paradigm. One such effort is the Codelet model which aims at providing scalable fine-grained execution for the upcoming exa-scale era. The Codelet model finds its inspiration in the dataflow and its descendants such as the hybrid dataflow/Von Neumann EARTH execution model [1].

In the codelet model, a program is represented as many codelets. Each codelet is a small piece of sequential codes. The codelets form a dependency graph (so called codelet graph) based on the data production and consumption relations among the codelets. During runtime, there is a codelet scheduler monitors and updates the dependency satisfactions on the codelet graph. Once a codelet has satisfied all its dependencies, the codelet scheduler will try to execute the codelet on an available processor. Comparing to the traditional coarse-grain models, the codelet model may achieve better resource utilization due to its finer granularity on scheduling.

[1]This work was supported by European FP7 project TERAFLUX, id. 249013 and also partially supported by Japan Society for the Promotion of Science (JSPS).

State-of-the-art codelet schedulers mainly focus on workload balancing. Once a processor is available, the scheduler will immediately schedule one ready codelet (if applicable) to it. Energy efficiency issue is left for programmers to study. In this paper, we propose a novel scheduling algorithm that targets on energy efficiency via automatic locality exploitation. Our major observation is as follows: If two codelets have potential locality (i.e., one produces some data and the other consumes the same data), then a codelet scheduler may exploit the potential locality by scheduling the two codelets on the same core. Then the producer codelet may store data on a local buffer (e.g., local scratchpad memory in each core of the IBM Cyclops-64 chip) for the consumer codelet to load. In such a way, the data is not necessary to go though global memory which normally consumes much more energy than the local storage.

Our scheduling algorithm firstly generates a static scheduling plan that assigns the codelets to the processors. During runtime, the codelet scheduler follows the scheduling plan to execute the codelets. The algorithm assumes that potential locality between every two dependent codelets is statically known. In most cases, this information can be easily obtained from programmer hints, compiler analysis, or profiling. The potential locality is represent as the weight of each edge in the codelet graph.

The algorithm is as follows: Initially, all the edges are put into an edge pool. Then the algorithm picks the max edge (i.e., the one with highest weight) from the pool. The two ends of the edge will be scheduled to the same processor in some adjacent execution order. The edges against the scheduling will be removed from the edge pool. The algorithm continues this process of picking and removing until the edge pool is empty.

The experimental result on our developed Cyclops-64 emulator shows that the algorithm reduces up to 59.7% of global memory accesses and 57.9% of energy consumption for randomly generated codelet graphs as well as matrix multiplication and merge sort.

References

[1] K. B. Theobald, "EARTH: an efficient architecture for running threads," Ph.D. dissertation, McGill University, Montreal, Que., Canada, Canada, May 1999, AAINQ50269.