An Energy Scalability Model for Efficient Resource Allocation on Manycore Architectures

Joosung Kim, Hakbong Kim, Hyunhee Kim, and Jihong Kim
School of Computer Science and Engineering
Seoul National University, Korea
Email: {testype, haknalgae, hh0726, jihong}@davinci.snu.ac.kr

***Abstract*—For energy-efficient manycore-based systems, energy scalability support is an important system requirement. In this paper, we propose an energy scalability model for dynamically multithreaded programs (written in Cilk) running on manycore-based systems. The proposed model takes advantages of execution characteristics of dynamically multithreaded programs based on a work-stealing scheduling model. Our experimental results show that the proposed energy scalability model can accurately estimate the energy consumption of Cilk applications under varying number of cores and threads within the average 5% of estimation errors.**

I. Introduction

For manycore-based systems with hundreds of cores, efficient scalability support is one of the most important requirements. For example, for a given parallel application *P*, understanding how *P*'s energy efficiency varies as the number of cores allocated to *P* changes is essential for the manycore-based systems to be managed in an energy-efficient fashion. Without such scalability information, a resource allocator/manager of the manycore-based systems cannot intelligently adapt to the dynamically varying resource requirements of various parallel applications. In this paper, we propose an energy scalability model for manycore-based systems as the number of allocated cores changes.

Although it is rather difficult to predict what programming languages would be widely accepted for programming such manycore-based systems, we conjecture that such programming languages would separate *how to express* the logical parallelism in a program from *how to map* the parallelism to cores. The latter task of mapping the parallelism to cores will be handled by a run-time system without a programmer's intervention. Furthermore, a run-time system should dynamically balance the workload among the cores. For these reasons, we choose Cilk [1], which is a high-level dynamically multithreaded language based on a work-stealing scheduling model, for this study.

More specifically, in this paper, we propose an energy scalability model for a Cilk parallel application in terms of the number of physical cores and the number of software threads (running on the physical cores). The proposed scalability model needs a prior execution time information and energy consumption information of a Cilk application when a single physical core and a single thread are used to run the Cilk application. Our model can be useful for a resource allocator/manager of the manycore systems to make an intelligent decision on the system's energy efficiency. For example, when a Cilk parallel application starts or terminates, the resource allocator/manager can adapt the core assignments to active Cilk parallel applications in a way that optimizes the overall energy efficiency of the system.

As a first step of our energy scalability study on manycore-based systems, we have developed an energy scalability model for ARM11 MPCore[1] which has four cores using the Cilk 5.4.6 system, and validated predicted energy consumption values with measured energy consumption values over different combinations of the number of physical cores and the number of software threads. Our initial results show that our model produces accurate energy estimates for most test cases with less than 5% error on average.

II. Energy Scalability Model

In the proposed energy scalability model, the energy consumption of a Cilk program *P* is computed by the product of the *average* power consumption of *P* and the execution time of *P*. In Cilk applications, a large number of fine-grained tasks are created during runtime and they are managed by the Cilk runtime system. Since the Cilk runtime is based on the *work-stealing* scheduling policy, these tasks tend to be evenly distributed to participating threads. Furthermore, these fine-grained tasks' power consumption characteristics are quite similar. Therefore, the power consumption of each thread, when a different number of cores and threads are used, can be reasonably estimated to be similar to that of each thread when a single core and a single thread is used to execute the threads.

As an input to our energy scalability model, we estimate the average power consumption of each thread under the combination of a single core and a single thread using the power model we have developed in our previous work [3].

In order to estimate the execution time of *P* under a different number of cores and threads, we extend the existing Cilk execution time scalability model [2] by better estimating the overhead cost when a different number of cores and threads are used. Cilk views the execution of a multithreaded program as a set of small work, *task*, and the parallel execution of a program can be described as a directed acyclic graph of tasks, *dag*. In this dag model, the execution time along the critical path is called *span*, which indicates the minimum possible execution time regardless of the number of cores and threads used.

The tasks which are not in the critical path can be distributed to multiple threads, and they can be concurrently executed.

[1]Although a four-core ARM11 MPCore is not a manycore processor, we think that it can be considered as a reasonable approximation of a part of the manycore processor. Since a single Cilk program is not likely to use all the available cores when there are other parallel applications that run concurrently, a single multicore can approximate a fraction of the manycore which was allocated to the single Cilk program, especially when allocated cores are spatially close.

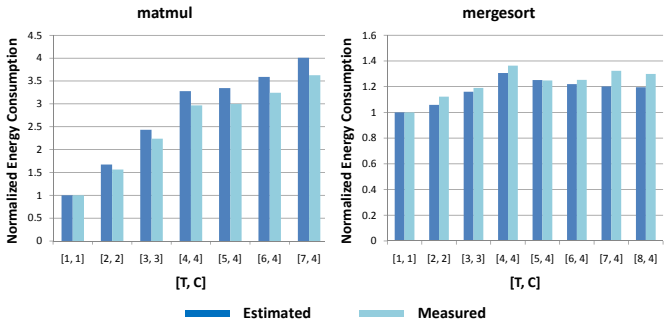

Fig. 1: Normalized Energy Consumption Comparisons

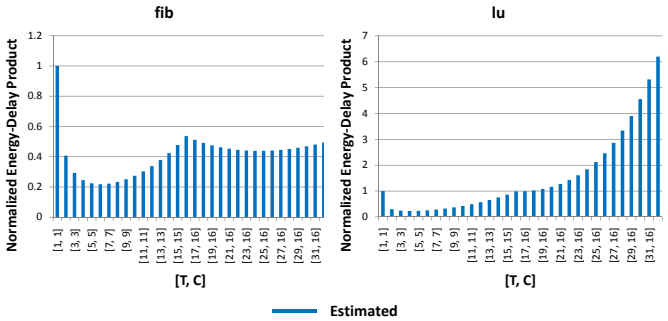

Fig. 2: Normalized Energy-Delay Product Estimations

TABLE I: Coefficients of The Overhead Model

Coefficient	Performance Event Type	Value
C_1	No. of L1 d-cache miss/L1 i-cache miss	$1.39E+01$
C_2	No. of shared L2 cache access	$3.42E+05$
C_3	No. of stall cycles due to data dependency	$-2.26E+07$

Since an actual execution time will include various overheads such as thread management cost, Cilk runtime overhead, and memory hierarchy misses, it is clearly higher than the span of the Cilk program.

In the extended model, we divide the execution time into three parts. For an application *P* with *C* cores and *T* threads, the execution time model is given as follows:

$$\begin{aligned} ExecTime(P,T,C) = & (ExecTime(P,1,1) - Span)/C \\ + & \ Span \\ + & \ Overhead(P,T,C) \end{aligned} \quad (1)$$

The first term of Eq. (1) estimates the execution time of *P* when the workload of *P* is evenly distributed among *C* cores and Cilk runtime-related overhead when *C* cores are used. The third term, $Overhead(P,T,C)$, can be further decomposed into three part as follows:

$$\begin{aligned} Overhead(P,T,C) = & (L1MRatio/C_1 \times (T-C))^2 \\ + & \ L2Access/C_2 \times C \\ + & \ DataDep/C_3 \times (T-C)/T \end{aligned} \quad (2)$$

Each term in Eq. (2) represents the effect of context switching and resource contention, inter-core communication, and task granularity, respectively. In the first term, the number of threads larger than that of physical cores increases execution time by interfering with each other. We select L1 data cache misses per L1 instruction cache miss (*L1MRatio*) to represent the degree of data communication overhead among threads. Moreover, due to work-stealing and synchronization characteristics of the Cilk runtime, uni-directional inter-core communications are necessary from spawned children to their parents. This linearly increases the overhead as the number of physical cores increases as in the second term. An L2 cache access event (*L2Access*) is selected to reflect the amount of data transfer and task migration along the memory hierarchy.

In the *work-stealing scheduler* of the Cilk runtime, an idle thread steals the remaining work from another thread, and thus the total work is evenly divided and distributed to threads. Because of this behavior, as the number of threads increases, the size of working sets of the threads decreases, which in turn improves the locality of the threads as represented in the last term of Eq. (2). Stall cycles due to data dependency (*DataDep*) is used to consider the reduction of workload size.

III. Experimental Results

Fig. 1 illustrates the accuracy of the proposed energy scalability model over the measured data under varying numbers of cores and threads. We estimate the energy consumption using the proposed model, and compare them to the measured data which are calculated by using actual event counter values and execution time of each cases. Over seven Cilk applications, the average estimation error was 4.99%.[2]

In Fig. 2, we evaluate how the proposed scalability model can be used in estimating the energy-delay product of a Cilk application when large cores are used. We assume that up to 16 cores can be allocated to a Cilk program while up to 32 software threads can be spawned. The result shows that different Cilk applications have different optimal configurations of cores and threads. For example, *fib* achieves the best energy-delay product when 6 cores and 6 threads are used while *lu* achieves the best energy-delay product when 4 cores and 4 threads are used.

IV. Conclusions

We proposed an energy scalability model for Cilk applications running manycore-based systems. The proposed scalability model accurately estimate the energy consumption of a Cilk application when different numbers of cores and threads are used. The model can also estimate the energy-delay product of a Cilk application. We plan to extend our initial work to a real manycore-based systems such as Intel's Single-Chip Cloud systems.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 20100018873, No. R33-2010-10095, and No. 2010-0020724). This work was also supported by the Brain Korea 21 Project in 2011. The ICT at Seoul National University and IDEC provided research facilities for this study.

References

[1] M. Frigo, C. E. Leiserson, and K. H. Randall, The Implementation of the Cilk-5 Multithreaded Language. In *Proceedings of International Symposium on Programming Languages Design and Implementation*, 1998, pp. 212-223.

[2] Y. He, C. E. Leiserson, and W. M. Leiserson, The Cilkview Scalability Analyzer. In *SPAA*, 2010.

[3] W. Choi, H. Kim, W. Song, J. Song, and J. Kim, ePRO-MP: A Tool for Profiling and Optimizing Energy and Performance of Mobile Multiprocessor Applications. In *Scientific Programming*, 17(4):285-294, 2009.

[2]The maximum estimation error was 6.97%, which happened in the Cilk fft application.