Energy-Efficiency Metric for Real-Time Monitoring of OpenMP Programs Executing on Multicore Systems

Maxime Mirka, Gilles Sassatelli, Abdoulaye Gamatié

Energy-Efficiency Metric for Real-Time Monitoring of OpenMP Programs Executing on Multicore Systems

Maxime Mirka, Gilles Sassatelli and Abdoulaye Gamatié
LIRMM (CNRS and University of Montpellier)
Montpellier, France
name.surname@lirmm.fr

Abstract

Energy-efficiency has been a major challenge in compute systems over the last decade. In this paper, we propose an approach for on-line energy-efficiency mesurement when executing OpenMP workloads on multicore systems. The novelty of our approach lies in the ability to monitor energy efficiency at run-time without prior knowledge of the application profile or code annotation. The solution relies on two new metrics: the Chunks per Second (CpS) and Chunks per Joule (CpJ). The former captures the quantity of work achieved by threads per unit time (i.e. a performance indicator). The latter indicates the quantity of work achieved by threads per unit energy, also corresponding to the performance per watt. We show that these new metrics are suitable information making it possible to perform energy efficiency analysis.

1. Introduction

Energy-efficiency has been a major challenge in compute systems over the last decade in both embedded and high-performance computing domains. By energy-efficiency, we mean the best compromise between execution performance and power consumption. In high-performance computing domain (HPC), it is often referred to by using the floating-point operation per second per watt (or FLOPS/W in short) metric. In embedded computing domain, the usual metrics are rather millions of instructions per second per watt (or MIPS/W in short). In both units the first components refer to the performance (i.e., FLOPS and MIPS) while the second reflects the power consumption. Based on the energy-efficiency metric, optimizing the energy-efficiency can be achieved. Existing approaches relying on various design techniques are already surveyed in literature [4].

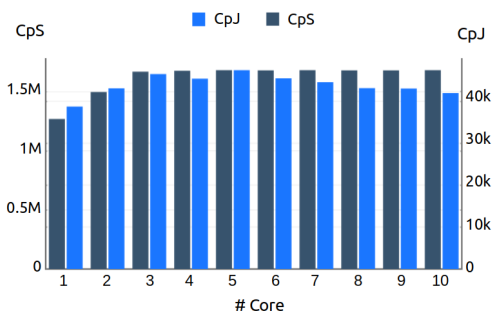
While the above techniques have been proven useful, the ability to exploit them in an adaptive way is central for optimized energy-efficiency. In particular, it is important to dynamically deal with application-specific energy consumption profile so as to take adequate decisions as early as possible. For this purpose, one usually relies on typical performance of power measurements that are informative-enough. Existing profiling tools such as PAPI [1] or scorep [2] can be used to gather the performance and power consumption numbers of a system. Such information are often obtained *a posteriori* i.e. after execution a large portion (if not the whole) of a given program, therefore, leaving only a little room for early optimizations. Analytic approaches, based on high-level estimation models for performance and power consumption, could be used for an earlier dynamic optimization. Unfortunately, such models are not necessarily reliable approximations for any execution platforms.

Contribution of this work. In this paper, we investigate a framework that enables real-time energy-efficiency measurement. Contrary to conventional approaches that either perform indirect measurement (i.e. on metrics that do not relate application performance directly) or require prior profiling (deriving energy efficiency from execution time), this approach taps into the OpenMP runtime and tracks application progress thereby enabling to derive energy efficiency. Our approach therefore applies to application programs written in OpenMP programming model which is by far the most popular shared-memory parallel API. It relies on two new metrics: *Chunks per Second* (CpS) and *Chunks per Joule* (CpJ). We show that these new metrics are suitable information allowing one to easily extract on-line programs' execution profile, opening opportunities for real-time optimizations. We further propose an extension of this work in [3].

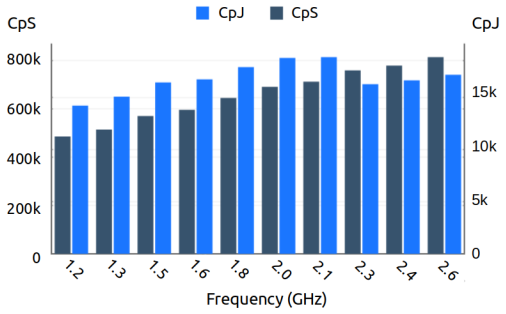
2. Energy-efficiency Characterization

OpenMP is the most popular parallel programming model for shared-memory systems. Its API supports C/C++ and Fortran programming languages among others on almost all architectures and operating systems, which makes it widely used.

OpenMP programs manage parallelism mostly by using threads. Threads are the smallest unit of processing that can be scheduled by an OS. A set of tools is provided by OpenMP to control parallelization and synchronizations. It is all based on two fundamental concepts: Fork and Join. The former is the transition from a sequen-



(a) Configurations with different core counts. Frequency = 1.2GHz.



(b) Configurations with different core frequencies (#core = 1).

Figure 1. CpS and CpJ for different system configurations.

tial region (master thread) to a parallel region (set of team threads) while the latter is the opposite transition, from a parallel to a sequential region.

Its features include parallel loops, in which jobs are partitioned in blocks of instructions, also called chunks. Chunks are assigned to the parallel threads previously defined during execution.

Light-Weight Metrics: In the following, we rely on this specific feature of OpenMP to define new light-weight metrics for characterizing both the performance and energy-efficiency of an application executing on multicore architectures: the Chunks per Second (CpS) and the Chunks per Joule (CpJ). The former is defined as the number of chunks executed in one second, where a chunk is a block of instruction assigned to a thread for execution. It denotes a speed of work, i.e., it is a performance metric. The latter defines the number of chunks executed per Joule, where the Joules designate the quantity of energy used by the compute system. It is also defined as CpS per Watt or CpS/W, the number of chunks per second executed per Watt. It denotes a quantity of work per quantity of energy, i.e., it is an energy efficiency metric.

The following example illustrates the characterization of a synthetic OpenMP code describing a memory-intensive program, using the new metrics.

Example: *We consider a memory-intensive application. When executed on various architecture configurations in terms of core count or operating frequencies, we obtain variable performance and energy-efficiency outcomes, through the obtained CpS and CpJ metrics, as shown in Figure 1. In Figure 1a, performance (i.e. CpS) increases up to 3 cores and reaches a plateau due to an obvious saturation of the memory subsystem given the rather memory intensive nature of the application. Having more than 3 cores further results in a decrease in energy efficiency as most cores are idling waiting for data to process whilst consuming power. When varying the freqency on a single core a monotonic increase in performance is observed in Figure 1b. Energy efficiency however drops past a certain frequency, possibly to rising contentions to the main memory.*

Implementation: Modifications are done on the compiler's OpenMP library, where the scheduling mechanisms responsible to dispatch chunks to threads are described. They consist in updating a chunk-counter saved in a shared memory, at any allocation of a chunk to a thread. Application binaries produced by this modified version of the OpenMP library therefore perform this automatic instrumentation. It then makes it possible to monitor CpS and CpJ values during application execution, by reading out from the shared memory region.

These metrics are easily implemented and suitable for a wide variety of architecture. As described previously, they can also be used at runtime, opening perspectives for online analysis of compute systems' efficiency.

3. Conclusion

On-line energy-efficiency measurement is essential for better management of resources in multicore systems. In this paper we focus on OpenMP programs, a parallel programming model widely used. We proposed two new metrics to evaluate both the performance and the energy-efficiency of compute systems during run-time.

This work opens new perspectives for energy consumption optimization. Indeed, the introduced metrics make possible on-line analysis of energy-efficiency. Further work could rely on such an analysis for adaptive systems, where adequate decisions will be taken upon the feedback from the CpS and CpJ analysis.

References

[1] S. Browne et al. A portable programming interface for performance evaluation on modern processors. *Int. J. High Perform. Comput. Appl.*, 14(3):189–204, Aug. 2000.

[2] D. Kai. Tools for assessing and optimizing the energy requirements of high performance scientific computing software. *PAMM*, 16:837–838, 2016.

[3] M. Mirka et al. Automatic energy-efficiency monitoring of openmp workloads. In *14th Int. Symp. on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2019.

[4] S. Mittal. A survey of techniques for improving energy efficiency in embedded computing systems. *IJCAET*.