Computers and Electrical Engineering 46 (2015) 95–111

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Balancing task- and data-level parallelism to improve performance and energy consumption of matrix computations on the Intel Xeon Phi ☆

Manuel F. Dolz [a], Francisco D. Igual [b,*], Thomas Ludwig [a], Luis Piñuel [b], Enrique S. Quintana-Ortí [c]

[a] Dept. of Informatics, University of Hamburg, 22.527 Hamburg, Germany
[b] Depto. de Arquitectura de Computadores y Automática, Universidad Complutense de Madrid, 28.040 Madrid, Spain
[c] Depto. de Ingeniería y Ciencia de Computadores, Universitat Jaume I, 12.071 Castellón, Spain

ARTICLE INFO

Article history:
Received 14 November 2014
Received in revised form 25 May 2015
Accepted 4 June 2015
Available online 19 June 2015

Keywords:
Power-aware computing
High performance
Many-core architectures
Runtime task schedulers
Dense linear algebra

ABSTRACT

The emergence of new manycore architectures, such as the Intel Xeon Phi, poses new challenges in how to adapt existing libraries and applications to this type of systems. In particular, the exploitation of manycore accelerators requires a holistic solution that simultaneously addresses time-to-response, energy efficiency and ease of programming. In this paper, we adapt the SuperMatrix runtime task scheduler for dense linear algebra algorithms to the many-threaded Intel Xeon Phi, with special emphasis on the performance and energy profile of the solution. From the performance perspective, we optimize the balance between task- and data-parallelism, reporting notable results compared with Intel MKL. From the energy-aware point of view, we propose a methodology that relies on core-level event counters and aggregated power consumption samples to obtain a task-level accounting for the energy. In addition, we introduce a blocking mechanism to reduce power and energy consumption during the idle periods inherent to task parallel executions.

1. Introduction

The performance of today's CMOS-based computer architectures is constrained by the cooling capacity of this technology and the power budget [1–3]. Concretely, the end of Dennard scaling [4] in the middle of the past decade marked the end of the "GHz race", and the shift towards multicore designs due to their more appealing performance-energy balance. Since then, the doubling of transistors on chip with each new semiconductor generation, dictated by Moore's law [5], has only exacerbated the problem [6].

In response to the *power wall*, many high performance computing (HPC) facilities have deployed heterogeneous clusters, equipped with manycore accelerators, such as AMD or NVIDIA graphics processor units (GPUs) or the Intel Xeon Phi, due to their favorable energy-performance balance as well as excellent acceleration for many compute-intensive applications.

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Associate Editor Dr. Jesus Carretero.
* Corresponding author.
E-mail addresses: manuel.dolz@informatik.uni-hamburg.de (M.F. Dolz), figual@ucm.es (F.D. Igual), thomas.ludwig@informatik.uni-hamburg.de (T. Ludwig), lpinuel@ucm.es (L. Piñuel), quintana@icc.uji.es (E.S. Quintana-Ortí).

http://dx.doi.org/10.1016/j.compeleceng.2015.06.009
0045-7906/

Moreover, the introduction of parallel programming standards, such as CUDA, OpenACC and OpenCL, has further increased the appeal of accelerator technologies. Nevertheless, programming an heterogeneous platform consisting of one to several general-purpose multicore processors and one or more manycore accelerators is still a considerable challenge. The reason is that, in addition to facing the programming difficulties intrinsic to having to exploit an ample amount of hardware concurrency, in many cases the developer also has to cope with the existence of multiple memory address spaces.

SuperMatrix is the run-time embedded in the `libflame` library [7] for the execution of dense linear algebra (DLA) operations on multicore desktop servers [8], heterogeneous CPU–GPU systems [9,10], and small-scale clusters [11]. The SuperMatrix run-time follows the methodology advocated in the FLAME project, which patronizes a separation of concerns between the derivation of new algorithms for DLA operations, their practical coding (implementation), and their high-performance execution on a given platform. SuperMatrix orchestrates a seamless, task-parallel execution of the full functionality of the `libflame` DLA library [7].

In this paper, we present an extension of SuperMatrix specifically tailored to tackle the considerable amount of hardware concurrency in the many-threaded Intel Xeon Phi processor. In doing so, our paper makes the following contributions:

- We adapt SuperMatrix to the Intel Xeon Phi manycore/many-threaded accelerator, demonstrating the benefits of abstracting the design and implementation of DLA algorithms from their practical, architecture-aware high performance execution. For our particular scenario, we rely on the *native programming model*, considering the accelerator as a stand-alone platform, in charge of the execution of both the runtime and the computational workload. With a research trend pointing in the direction of integrating accelerators and conventional architectures into the same chip, we envision an scenario where the accelerator becomes the main processor and, therefore, the native programming model is natural.
- We investigate the impact on performance of exploiting the concurrency implicit to DLA operations at two different levels: as task-parallelism only, exposed by the run-time, or from a combination of task- and data-parallelism, with the latter extracted via a multi-threaded implementation of the BLAS (*Basic Linear Algebra Subprograms*).
- From the perspective of energy efficiency, we describe a methodology that relies on core-level event counters and aggregated power consumption samples from the complete accelerator to deliver a detailed accounting of the energy dissipated by a DLA operation at the granularity of individual tasks.
- We illustrate the positive effect on energy consumption of modifying the SuperMatrix run-time to adopt an idle-wait (blocking) approach for idle threads instead of the conventional power-hungry busy-wait (polling).
- We provide an experimental evaluation and validation of the above contributions using a key DLA kernel, the Cholesky factorization, representative of the parallelism exhibited by many kernels in BLAS and LAPACK.

The rest of the paper is structured as follows. In Section 2 we further motivate our work, we review several related efforts, and we clarify the differences between these and our approach. In Section 3, we briefly present the SuperMatrix run-time system using the Cholesky factorization as a workhorse case study. In Sections 4 and 5, we describe and evaluate the different approaches that are employed to exploit the hardware concurrency of a 60-core Intel Xeon Phi 5110P accelerator. At this point, we note that our approach exploits the parallelism present at two different levels: at the task level via worker threads of the SuperMatrix run-time and at the data-parallel level via BLAS threads. (In addition, the codes implicitly exploit the SIMD parallelism of the floating-point units, or FPUs, of the Intel Xeon Phi by means of the vector operations embedded inside tuned implementations of the BLAS.) In Section 6, we introduce our two main energy-related contributions, namely the methodology to conduct a task-level energy accounting and the evaluation of an energy-aware scheduler. Finally, the paper is closed with a few concluding remarks in Section 7.

2. Motivation and related work

2.1. High performance and programmability

In recent years, a number of *run-times* have been proposed to alleviate the burden of programming multi- and many-threaded platforms. Concretely, OmpSs,[1] StarPU,[2] and Harmony,[3] among others, offer implicit parallel programming models with dependence analysis. When applied to the DLA operations that lie at the bottom of the "food chain" of many scientific compute-intensive applications [13], SMPSs (a precursor of OmpSs) [14], StarPU, Quark [15] and SuperMatrix [12] have demonstrated the advantage of extracting *task-parallelism* using this "run-time approach". Basically, all these software efforts exploit the task-parallelism implicit to the DLA operation by (semi-) automatically decomposing the operation into tasks while simultaneously performing a task-dependency analysis. This process is complemented with a dependency-aware out-of-order scheduling of the tasks to the computational resources at execution.

Among the run-time-based solutions, OmpSs, StarPU, Kaapi and SuperMatrix have all been ported to a heterogeneous platform equipped with multicore processors and one or more GPU accelerators. OmpSs and Kaapi have also been evaluated

[1] http://pm.bsc.es/ompss/.
[2] http://runtime.bordeaux.inria.fr/StarPU/.
[3] http://code.google.com/p/harmonyruntime/.