Experimental Study of Matrix Multiplication on MultiCore Processors

P. D. Michailidis, K. G. Margaritis

Matrix computations is one of the most important computational tasks in scientific computing. Two well-known parallel versions are Cannon's algorithm and the ScaLAPACK outer product algorithm. Typically, parallel implementations work well on 2D processor grids: input matrices are sliced horizontally and vertically into square blocks; there is a one-to-one mapping of blocks onto physical resources; several communications can take place in parallel, both horizontally and vertically. Even better, most of these communications can be overlapped with (independent) computations. All these characteristics render the matrix product kernel quite amenable to an efficient parallel implementation on 2D processor grids.
However, algorithms based on a 2D grid (virtual) topology are not well suited for multicore architectures. In particular, in a multicore architecture, memory is shared, and data accesses are performed through a hierarchy of caches, from shared caches to distributed caches. We need to revisit the parallel implementations of matrix computations in the context of data partitioning in order to improve the parallel execution on multicore architectures. This work present three parallel implementations of two fundamental matrix computation kernels such as matrix - vector multiplication and matrix multiplication on multicore processors. These parallel implementations are based on the three approaches of data partitioning among the available processing units such as row-wise, column-wise and square block-wise decomposition. Furthermore, for each possible parallel implementation is analyzed experimentally using the OpenMP programming environment on a machine with 4 Intel dual-core processors - a total of 8 cores. More specifically, we examine the performance results (i.e. execution times and speedups) of the proposed parallel implementations for matrix sizes ranging from 200x200 to 5000x5000 and different number of threads ranging from 2 to 8.
Also, this work we study how to model the performance of the proposed implementations to multicore architectures by taking memory access costs into account. More specifically, we present our effort to quantify and model each parallel implementation according to a performance model. The performance model of an implementation depends on two main aspects: the computational cost and the communication cost. In the case of multicore, communications are performed through direct read/write operations in a common memory address shared by two or more threads. To determine the computational cost we use an analytical model based on the number of operations and their cost in CPU cycles. Similarly, to predict the communication cost we model the memory access times using different methods according to its behavior - linear or non-linear. The proposed performance model validates against experimental results and it shows that the model is able to predict the parallel performance slightly even if the general behavior is correct. Finally, the proposed analytical prediction model can be used to predict the performance of two matrix computations for any problem size

(i.e. matrix size), number of available processors/cores and processor characteristics.