

Enhancing Muesli's Data Parallel Skeletons for Multi-Core Computer Architectures

Philipp Ciechanowicz and Herbert Kuchen

University of Münster

Leonardo-Campus 3

48149 Münster, Germany

Email: {ciechanowicz|kuchen}@wi.uni-muenster.de

Abstract—Algorithmic skeletons encapsulate typical parallel programming patterns such that they can be easily applied by users. Existing skeleton libraries usually work on distributed memory machines. We present an extension of our skeleton library Muesli which now allows to use the same application without modifications on a variety of parallel machines ranging from multi-processor distributed memory to many-core shared memory machines and combinations of those such as clusters of multi-core nodes. Internally, the skeletons are based on MPI and OpenMP. We demonstrate the efficiency of our approach by providing experimental results.

Keywords—distributed computing; distributed memory systems; parallel programming; programming environments; message passing; multiprocessing; shared memory systems;

I. INTRODUCTION

Writing a parallel program for a multi-processor computer can be a tedious and complicated task. When using communication libraries such as MPI directly, the abstraction level is rather low. The programmer has to think about decomposing the problem, integrating the partial solutions, and bother with communication problems such as deadlocks and starvation. More recently, multi-core processor architectures have evolved. While this provides additional parallelism, it also increases the complexity of writing efficient parallel programs. Although it is quite clear how to make use of multi-processor architectures, this is not yet the case for a cluster of multi- or many-core nodes. In order to reduce complexity and simultaneously come up with a higher level of abstraction, *algorithmic skeletons* have been proposed [1]. They provide predefined parallel computation and communication patterns and hide the parallelism from the user.

The main contribution of this paper is to present a substantial extension of the Muenster Skeleton Library Muesli [2] in terms of new data parallel skeletons for all our distributed data structures. These new skeletons now efficiently take advantage of recent multi-core computer architectures by simultaneously using the Open Multi-Processing API OpenMP [3] in combination with MPI. According to the main idea of a skeleton library, i.e. hiding parallelism from the user, the support of multi-core computers is completely transparent to the user. Thus, not a single line of source code has to be changed when porting a program to a multi-core platform. Instead, it only has to be recompiled, the rest is taken care of inside our library.

Besides discussing the underlying programming model we will show implementation details and demonstrate the efficiency of our implementations by presenting a couple of benchmarks.

The remainder of this paper is structured as follows: Section II briefly introduces the Muenster Skeleton Library. Then, Section III describes our unique parallelization concept and its properties in detail. Section IV introduces the OpenMP Abstraction Layer used to implement our flexible approach. Afterwards, the `foldColumns` skeleton implemented for the class `DistributedSparseMatrix` is presented in Section V. Resulting benchmarks are shown in Section VI, related work is covered in Section VII. Finally, Section VIII concludes and gives an outlook to future work.

II. THE MUENSTER SKELETON LIBRARY MUESLI

The Muenster Skeleton Library Muesli [2] is a C++ template library making use of MPI 1.2 [4] and OpenMP 2.5¹ [3] to efficiently support multi-processor and multi-core computer architectures and combinations of those. It has been developed to relieve programmers of parallel applications from low-level programming problems by offering predefined parallel computation and communications patterns, so-called *algorithmic skeletons*. These skeletons can roughly be divided into task and data parallel ones. Task parallel skeletons are used to construct skeleton topologies and are offered as separate classes. Currently, Muesli provides task parallel skeletons such as `Pipe`, `Farm` [5], `DivideAndConquer` [6], and `BranchAndBound` [7]. Data parallel skeletons such as `fold`, `map`, `scan`, `zip`, and their variants are used to manipulate elements of a distributed data structure in parallel and are offered as member functions. Currently, Muesli provides distributed data structures for arrays, matrices, and sparse matrices [8]. Analogously to P³L [9], task and data parallel skeletons can be nested. Since Muesli is a template library, all skeletons offer a template parameter to flexibly define the data type used by the skeleton. As unique features, Muesli provides support for partial functions and currying [10] and an automated serialization mechanism inspired by [11] such that skeletons can exchange arbitrary data types, i.e. integrated and user-defined ones.

¹Unfortunately, OpenMP 3.0 is not available on our test platform.