August 2017

Towards Distributed Parallel Programming Support for the SPar DSL

Dalvan Griebler [a,1], Luiz Gustavo Fernandes [a]
[a] *Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil*

Abstract. SPar was originally designed to provide high-level abstractions for stream parallelism in C++ programs targeting multi-core systems. This work proposes distributed parallel programming support for SPar targeting cluster environments. The goal is to preserve the original semantics while source-to-source code transformations will be turned into MPI (Message Passing Interface) parallel code. The results of the experiments presented in the paper demonstrate improved programmability without significant performance losses.

Keywords. Parallel Programming Languages, Domain-Specific Language, Stream Parallelism, Algorithmic Skeletons, Parallel Patterns, Parallel Programming.

1. Introduction

Parallel programming for cluster architectures remains a challenging task for application programmers. For the most part, they need to deal with source code rewriting and low-level programming libraries when they try to achieve high-performance in distributed parallel processing. The current standard is the Message Passing Interface (MPI), which requires code modeling with explicit communication implementation, load balancing, processes synchronization, and data serialization (except MPI data types). Programmers must also understand the underlying architecture to efficiently exploit the parallelism.

This problem is well documented in the literature and though there are state-of- the-art initiatives such as the X10 [18], Chapel [5] and Charm++ [6], there is still a lack of high-level and productive alternatives such as proposed in SPar [13]. For instance, Charm++ is a machine independent programming system based on C++. The programmer exploits the parallelism in a object-oriented manner, and the runtime can run object and classes in parallel while the communication must be done through message passing [1]. On the other hand, X10 is a new object-oriented parallel programming language designed to work with the Asynchronous Partitioned Global Address Space (APGAS) model [15]. Moreover, targeting heterogeneous parallel architectures (CPU and GPU), there are related approaches that use C++ attributes (the same annotation mechanism of SPar) to abstract parallelism as part of a software engineering methodology [8,9]. However, they do not provide support for distributed parallel programming.

In contrast to these initiatives, we are proposing support for distributed parallel programming without changing the original semantics of SPar. In fact, our approach enables application programmers to add standard C++ annotations rather than having to rewrite their sequential code on shared memory architectures (multi-core) [12,11]. Our goal in

[1] Corresponding Author: dalvan.griebler@acad.pucrs.br