T++: an Object-Oriented Language to Express Task and Data Parallelism on Multi-SIMD Computers

Marc Michel PIC, Hassane ESSAFI, Marc VIALA and Laurent NICOLAS

LETI (CEA – Technologies Avancées)
DEIN – CE/S F91191 Gif-sur-Yvette Cedex.
phone: (+33) (1) 69 08 53 35
mpic@chouette.saclay.cea.fr

Abstract. In this paper we introduce T++: a parallel language with object-oriented features designed for Multi-SIMD parallel computers. We propose a new approach to express simultaneously task and data parallelism. We describe the advantages of an object-oriented approach and what kind of semantics we choose to structure our task-data-parallelism. Finally, we explain how to implement it efficiently on a proprietary Multi-SIMD architecture: the SYMPHONIE concept.

Keywords: task-parallelism, data-parallelism, massively parallel systems, language constructs, semantics, Multi-SIMD computers, scientific computing.

1 T++ project: Introduction

The goal of this project is to design an high-level language for Multi-SIMD computers. A Multi-SIMD architecture is a compound machine of several (heterogeneous or homogeneous) SIMD computers. We call each SIMD architecture a node. Each node has its own Control Unit, and so could execute its own program. Each node consists of one or more Processing Elements (PEs)\(^1\). In the first part we introduce SYMPHONIE, a Multi-SIMD architecture designed in our laboratory for Real-Time Image Processing and High-Performance Physics Simulations. In the second part, we describe the high-level language T++ based on C++, which allows us to program such architectures using an extension of the data-parallel paradigm. We retain the philosophy of array programming languages, to which we add tools to express multitasking. These tools consist of arrays of tasks. Each task array defines a set of operations to be executed concurrently on a parallel data array. In the third part, we explain the implementation techniques we use to prepare a compiler for this language.

\(^1\) MIMD computers are included in this description and our language could be used to program them, but it is designed more specifically for architectures with a number of PEs larger than the number of nodes.
2 The SYMPHONIE Computer

2.1 Description

SYMPHONIE is a Multi-SIMD computer where each SIMD node is a linear array processor. SYMPHONIE is conceived especially for image processing but matrix computation applications can be performed efficiently. Each node contains from 32 up to 1024 PEs managed by one Control Unit. Each PE consists mainly of 32 bits processor, a coprocessor dedicated to memory address computation, a floating point accelerator and a communication module. The SIMD nodes are connected using asynchronous intercommunication system. The same system (patented by CEA-LETI) is also used at the SIMD level to perform irregular communication between the PEs[CEJK92]. This system is necessary to SIMD structure for Image Processing. It is designed by the LETI (CEA) for Real-Time Image Processing. It follows the family of SYMPATI SIMD computers designed by CEA-LETI in collaboration with IRIT (University of Toulouse) and produced by Centralp.

2.2 Performances

Performances of SYMPHONIE have been evaluated on specific image processing codes. Its performance for 4 nodes of 1024 PEs is 160 GigaOPS. Table 1 illustrates the performances of one node of 128 processors (one VME card):

<table>
<thead>
<tr>
<th>Operations on (256×256) images</th>
<th>Time (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hough Transform</td>
<td>2.2</td>
</tr>
<tr>
<td>Histogram Equalization</td>
<td>0.3</td>
</tr>
<tr>
<td>Labelling</td>
<td>2.5</td>
</tr>
</tbody>
</table>

3 Main Concepts

3.1 A Language for Task and Data Parallelism

Our motivation for the concept of T++ is to extend the data-parallelism paradigm to manage simultaneously the task parallelism.

The main point in the philosophy of this language is that Multi-SIMD computers can be considered as MIMD computers at large scale and as SIMD computers locally. Moreover our language divides the ambivalence of those architectures into two concepts of programmation: the first concept is dedicated to the SIMD level (i.e., the data-parallel one), and the second is dedicated to the MIMD