VSIPL++: Serial and Parallel Performance

Mark Mitchell
Jeffrey Oldham
Nathan Sidwell
CodeSourcery, LLC

May 22, 2003

1 Introduction

The VSIPL (the Vector, Signal, and Image Processing Library) specification defines a portable, C programming language interface to use in linear algebra and signal-processing applications. The VSIPL standard has been implemented by a variety of vendors. VSIPL's portable interface provides developers the ability to write code once and reuse it in multiple environments.

At HPEC 2002, we presented an overview of VSIPL++, a C++ specification designed to perform the same types of computations as VSIPL. The primary goals for VSIPL++ are improved serial performance relative to VSIPL, support for multi-processor systems, extensibility, and simpler syntax.

The serial VSIPL++ specification is virtually complete. By HPEC 2003, we expect to have a successful implementation of the specification. We anticipate that the performance of the VSIPL++ reference implementation will be superior to that of VSIPL for some applications. By HPEC 2003, the reference implementation of VSIPL++ will contain preliminary support for parallel systems.

Our presentation will compare the performance of VSIPL++ with VSIPL, and demonstrate the VSIPL++ support for parallel computation. We will also discuss VSIPL++ implementation strategies, including the use of an existing VSIPL implementation, a native C++ implementation using expression-templates, and a hybrid approach that allows an implementor to incrementally reimplement portions of VSIPL++ to achieve higher performance.

2 Performance Comparisons

VSIPL++ can be implemented on either uni-processor or multi-processor hardware. As a first step, we are implementing VSIPL++ using an existing C VSIPL library. While this implementation is straightforward, the performance is of course limited by the performance of the underlying VSIPL implementation. We also have a preliminary implementation of some portions of VSIPL++

using a high-performance expression-template technique. By HPEC 2003, we plan to have a partial parallel implementation of VSIPL++.

We are using a simple FIR-filter and narrowband beamforming application as a benchmark. These computations are fundamental to many signal-processing applications. We plan to present performance comparisons between VSIPL, VSIPL++ built atop VSIPL, VSIPL++ using expression templates, and VSIPL++ using multiple processors.

3 Parallel Computation Model

VSIPL++ uses a Single Program Multiple Data (SPMD) model when performing parallel computations. The VSIPL++ model divides rectangular arrays of data (known as "blocks") into sections using combinations of block and cyclic data distributions. We will explain the VSIPL++ model, and demonstrate how a very simple distribution model can accommodate systems ranging from small embedded systems to large systems with thousands of nodes. We will also explain how a wide variety of distribution policies can be implemented atop the simple distribution model provided by VSIPL++. Finally, we will explain how the VSIPL++ parallelism model provides support for fault-tolerance via dynamic reallocation of processors.