VSIPL++: Intuitive Programming Using C++ Templates

Mark Mitchell Jeffrey D. Oldham

2002 Aug 22

Submission Information

Author Information:

Mark Mitchell
CodeSourcery, LLC
9978 Granite Point Court
Granite Bay, CA 95746
916.791.8304 (voice)
916.914.2066 (fax)
mark@codesourcery.com
citizen of USA

Jeffrey D. Oldham
CodeSourcery, LLC
938 Clark Ave #34
Mountain View, CA 94040
650.968.0708 (voice)
916.914.2066 (fax)
oldham@codesourcery.com
citizen of USA
first, corresponding, presenting author

U.S.-Only Session: no

Poster: no

Topic Areas:

- software architectures, reusability, scalability, and standards
- algorithm mapping to high performance architectures
- middleware libraries and application programming interfaces

Abstract

VSIPL++ is a high-performance C++ toolkit for vector and signal processing applications. Building on the successful Vector, Signal, and Image Processing Library (VSIPL) standard (a C standard for similar applications), VSIPL++ adds additional features including:

- direct support for parallel computation,
- simpler syntax, improved type-checking, and other improvements to reduce validation and verification (V&V) costs,
- support for specialized data storage formats, and
- potentially higher performance.

Programs built with VSIPL++ will be as fast or faster as hand-coded VSIPL programs, but require less effort to build and will automatically work in both serial and parallel environments. VSIPL++ includes all of VSIPL's functionality, i.e., vector and matrix operations, signal processing functions, and linear algebra.

VSIPL++ uses C++ to reduce the burden on programmers. For example, a matrix `m` is created by invoking the `Matrix` constructor, which automatically handles all necessary memory allocation.

VSIPL++ also directly supports data-parallel programming. For example, a matrix having entries equal to the cosine of matrix `m`'s entries is denoted `cos(m)`. The sum of two matrices can be assigned to a third using a data-parallel assignment `m0 = m1 + m2`, but the logically invalid sum of a `Vector` and a `Matrix` yields a compile-time error. The same syntax can be used to manipulate scalars, vectors, matrices, and tensors.

In VSIPL++, in contrast to VSIPL, data may be stored in formats other than simple arrays of contiguous memory. Programmers use logical *views* of data. These views are manipulated as if they were contiguous arrays. Choosing an alternative storage format does not require the programmer to change the code that manipulates the memory. In addition, programmers may superimpose more than one view on the same data. For example, a programmer can have both a one-dimensional (vector) view and a two-dimensional (matrix) view of the same data. Alternative storage formats can reduce the use of memory (e.g., by storing only non-zero values), improve performance (e.g., by computing values lazily), or improve reliability (e.g., by replicating values and then using voting to determine the value stored).

VSIPL++ supports distributed computation using the single-program, multiple data model. Programmers specify data distribution when containers are created, but all other portions of the program are written the same for serial and parallel programs. VSIPL++ automatically distributes the computations across processors. This style of programming greatly reduces validation and verification costs by permitting debugging on serial machines and by confining the complex, error-prone task of writing parallel code to the toolkit itself.

VSIPL++ makes use of C++ templates to provide intuitive syntax while generating highly efficient code. For example, the use of novel data storage formats does not impose any additional overhead on programs that use traditional data formats, nor do users of the novel data storage formats pay additional costs relative to hand-coded programs using the traditional data formats.

A draft version of the VSIPL++ specification is available and work on a reference implementation is underway. We anticipate that the complete specification and a complete reference implementation will be available by the end of 2003.