Development of a Stencil Compiler for One-Dimensional Convolution Operators on the CM-5

Roch Bourbonnais
Thinking Machines Corporation,
245 First Street, Cambridge, MA 02142-1214, USA

Abstract. Finite difference methods are ideally suited to modern computer architectures and in particular parallel ones. Convolution operators or "stencils" are commonly used to solve a wide variety of problems. These operators can be elegantly written in data-parallel language but the performances obtained can be lower than expected.

With its cache-less vector node architecture, the CM-5 is particularly well suited to this type of operation on large volumes of data. We have developed a code generator that can improve performance over Connection-Machine Fortran (CMF) by a factor of sometimes 3 and reach computation rates in excess of 100 Mflops/node.

We will first describe the performance characteristics of stencil operators expressed in Connection Machine Fortran. Then, based on a performance model of the CM-5 processors, we will describe the three vectorization strategies that were implemented in the code generator.

1 Introduction

1.1 Justification

Finite difference methods are ideally suited to modern computer architectures and in particular parallel ones. In many cases, the integration scheme can be written in terms of convolution operators in which the source data are shifted then multiplied by a coefficient. These operators are collectively called "stencils".

In the case of petroleum reservoir modelling, it is common to see stencil expressions with a large (≥ 6) number of terms and in which the source data are shifted along a single axis of the 2-D or 3-D data set [1, 2, 3].

Despite the fact that these operators can be simply and elegantly written in data-parallel language such as HPF, the performances obtained are usually quite far from peak performance of the machines. The problem is that data-parallel constructs introduce extra temporaries and data-movement which are not strictly required to implement the stencils. Current compilers are unable to recognize stencils or produce inefficient code.

With its cache-less vector node architecture, the CM-5 is particularly well suited to this type of operation on large volumes of data. We have developed a code generator that can improve performance over Connection-Machine Fortran
(CMF) by a factor of sometimes 3 and reach computation rates in excess of 100 Mflops/node. The low-level CDPEAC (vector units assembler) generated code minimizes the within processor data movements keeping only the inevitable off-processor communication.

To reach good performance on a variety of cases, we developed three underlying strategies, each one striding memory in a particular order. The fastest vectorization strategy is selected at run-time using a semi-heuristic method that takes into account the data distribution in processor memory.

This work has led to the creation of a stencil utility capable of generating code for one-dimensional stencils of up to 9 points in either single or double precision, with either scalar or array coefficients.

This work distinguishes itself from other efforts, notably from a stencil compiler developed in the framework of the Connection Machine Scientific Software Library [4] or other [5], in that it aims to provide the highest possible performance reachable on a CM-5 system but only on a restricted set of stencils to be defined in the following section.

After the definitions, we will be discussing the implications of writing stencils using a high level language such as Connection-Machine Fortran [6]. We will then discuss in detail the implementation of our stencil generator for the CM-5.

1.2 Definitions

A stencil is formed by the successive accumulation into a destination array of a number of terms or points where each term is the product of a coefficient with a CSHIFT-ed [7, 6] source array. Each coefficient can be either full size arrays with the same rank and dimensions as the source data or more simply a scalar value. We differentiate variable (or array) coefficients to fix (or scalar) coefficient stencils. One-dimensional stencils are characterized by the fact that all terms of the expression are shifted relative to one another along what is called the stencil axis. A compact one-dimensional stencil is one in which the successive terms involve shifted data with consecutive shift counts. To specify an n-point one-dimensional compact stencil, one also needs to give the starting shift value which we call the from parameter; For example a 5 point compact stencil in which the first shift count is -3 will be formed of terms with shift counts of -3,-2,-1, 0, and 1 (see fig. 1).

If n is odd, then the stencil can be centered which means that for each term with a negative shift (−s) value there will be a term with the positive shift value (+s). So from here on, we will consider a one-dimensional compact stencil to be uniquely specified at compile time by :

\[ \text{n} \quad \text{:} \quad \text{the number of terms or points in the expression,} \]
\[ \text{from} \quad \text{:} \quad \text{the leftmost (or smallest) shift value,} \]
\[ \text{prec} \quad \text{:} \quad \text{the precision of all floating point data involved,} \]
\[ \text{coeff type} \quad \text{:} \quad \text{the kind of coefficients used (fix or variable).} \]

Furthermore, at runtime, it will be necessary to specify :