Data-Parallel Compiler Support for Multipartitioning⋆

Daniel Chavarría-Miranda, John Mellor-Crummey, and Trushar Sarang

Dept. of Computer Science MS132, Rice University
Houston, TX 77005
\{danich,johnmc\}@cs.rice.edu

Abstract. Multipartitioning is a skewed-cyclic block distribution that yields better parallel efficiency and scalability for line-sweep computations than traditional block partitionings. This paper describes extensions to the Rice dHPF compiler for High Performance Fortran that enable it to support multipartitioned data distributions and optimizations that enable dHPF to generate efficient multipartitioned code. We describe experiments applying these techniques to parallelize serial versions of the NAS SP and BT application benchmarks and show that the performance of the code generated by dHPF is approaching that of hand-coded parallelizations based on multipartitioning.

1 Introduction

High Performance Fortran (HPF) and OpenMP provide a narrow set of choices for data and computation partitioning. While their standard partitionings can yield good performance for loosely synchronous computations, they are problematic for more tightly-coupled computations such as line sweeps. Line sweep computations are the basis for Alternating Direction Implicit (ADI) integration—a widely-used numerical technique for solving partial differential equations such as the Navier-Stokes equation [5,10], as well as a variety of other computational methods [10]. Recurrences along each dimension of the data domain make this class of computations difficult to parallelize effectively.

To support effective parallelization of line-sweep computations, a sophisticated strategy for partitioning data and computation known as multipartitioning was developed [5,10]. Multipartitioning distributes arrays of two or more

⋆ This work has been supported in part by NASA Grant NAG 2-1181, DARPA agreement number F30602-96-1-0159, and the Los Alamos National Laboratory Computer Science Institute (LACSI) through LANL contract number 03891-99-23, as part of the prime contract (W-7405-ENG-36) between the Department of Energy and the Regents of the University of California. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied of sponsoring agencies.
dimensions among a set of processors so that for computations performing a line sweep along any one of the array’s data dimensions, (1) all processors are active in each step of the computation, (2) load-balance is nearly perfect, and (3) only a modest amount of coarse-grain communication is needed. These properties are achieved by carefully assigning each processor a balanced number of tiles between each pair of adjacent hyperplanes that are defined by the cuts along any partitioned data dimension. Figure 1 shows a 3D multipartitioning for 16 processors; the number in each tile indicates the processor that owns the block. For 3D problems, “diagonal” multi-partitionings [10] can be applied when $\sqrt{p}$ is integral, where $p$ is the number of processors. This strategy involves partitioning the data domain into $p^2$ tiles. Each processor handles $\sqrt{p}$ tiles arranged along diagonals through the data domain. Recently, we developed an algorithm for applying multipartitionings efficiently on an arbitrary number of processors, which significantly broadens their applicability [7].

A study by van der Wijngaart [11] of implementation strategies for hand-coding parallelizations of Alternating Direction Implicit Integration (ADI) found that 3D multipartitioning was superior to both static block partitionings with wavefront parallelism, and dynamic block partitionings in which each phase of computation is perfectly parallel but data is transposed between phases.

Our earlier research on data-parallel compilation technology to support effective, semi-automatic compiler-based parallelizations of ADI line-sweep computations focused on exploiting wavefront parallelism with static block partitionings by using a coarse-grain pipelining strategy [1]. Although the performance we achieved with this approach was superior to that achieved for a version of the codes using dynamic block partitionings compiled with the Portland Group’s pgHARF compiler [4], both of the compiler-based parallelization strategies fell significantly short of the performance achieved by hand-coded parallelizations of the applications based on multipartitioning [1].

To closer approach the performance of hand-coded line-sweep computations with compiler-based parallelizations, we have been developing data-parallel compiler support for multipartitioning. A previous paper [6] describes basic compiler and runtime techniques necessary to support multipartitioned data distributions and a prototype implementation in the Rice dHPF compiler. Measurement of

Fig. 1. 3D Multipartitioning on 16 processors