REFERENCIAS BIBLIOGRÁFICAS


Compiler Managed Cache

Edward M. Robinson
Binary Consulting, Inc.
Gaithersburg, MD, USA

Ernst L. Leiss
Dept. of Computer Science
University of Houston
Houston, TX, USA

Abstract

We describe our architecture for replacing virtual memory with a compiler and runtime system. Our approach to the problem is based on a compiler managed cache, where main memory is the cache and the program is restructured to automatically move data between main memory and disk. The compiler is responsible for directing the runtime system as to which data can be transferred back to disk and when.

Our system performs at least as well as virtual memory management (VMM) in all test cases and significantly better in most cases. Specifically, our system performs, on average, 1.9 times faster than VMM for single process execution and 5.5 times faster for four simultaneous processes.

Keywords: Compiler optimization, I/O minimization, Out-of-core performance

1 Overview

Our architecture (see Figure 1) is part of a complete programming process where the programmer works to create an efficient I/O minimal program. The process of building I/O minimal programs is divided into several stages. At the core of the architecture is an extensible internal format which is used to represent programs. The format has primitives for operations on scalars, array access, and record access. The format represents a hierarchical data space with controlled access to array and record elements. Our format is called UHF, which stands for universal hierarchical format.

Legacy Fortran and C programs require special consideration. Programs that reshape arrays or cast pointers must be carefully handled to avoid inefficient, least common denominator translation. Hence, an initial migration phase is inserted to allow programmers to assist the compiler in converting programs into the UHF format. Optimizations can be applied during the migration phase to enhance the quality of the code. For example, a Fortran do loop can be transformed into a doall loop if it satisfies the dependence requirements.

Further transformations can be applied to the internal format and viewed using visual tools. Every aspect of a program is stored in a collection of persistent objects. These objects are manipulated by other objects or directly by a graphical user interface (GUI). This allows extension of semantics of the programming model without the difficulty of extending the syntax of a specific