Exploring Memory Access Regularity in Pointer-Intensive Application Programs

Keqiang Wu, Resit Sendag, and David J. Lilja

Department of Electrical and Computer Engineering, University of Minnesota
200 Union Street S.E., Minneapolis, MN 55455, USA
{kqwu, rsgt, lilja}@ece.umn.edu

Abstract. Pointer-intensive and sparse numerical computations typically display irregular memory access behavior. This work presents a mathematical model, called the Self-tuning Adaptive Predictor (SAP), to characterize the behavior of load instructions in procedures with pointer-based data structures by using procedure call boundaries as the fundamental sampling frequency. This model incorporates information about the history of specific load instructions (temporal locality) and their neighboring loads (spatial locality) using a least-squares minimization approach. Simulation results on twelve of the most time-consuming procedures with pointer-based data structures from five of the SPEC2000 integer benchmark programs show that these pointer-based data structures demonstrate surprisingly regular memory access patterns. The prediction error at steady-state is within [-6%, +6%] on average.

1 Introduction

An important characteristic of Pointer-Based Data Structures (PDS) is that they are dynamically allocated and managed using a heap. Heap allocation allocates blocks of contiguous memory as requested by the program at run-time. Memory blocks are deallocated in any order either explicitly or via process termination. For example, elements in a linked data structure contain explicit fields that name all adjacent elements by address. This mode of connectivity allows the easy construction and manipulation of data structures of arbitrary shape, such as trees and graphs. Dynamic construction allows PDSs to grow arbitrarily large. However, this flexibility makes it challenging to characterize the memory access behavior of these structures. Their behavior has been traditionally classified as irregular or arbitrary [1-2].

The intuitive method for prediction is to track the memory allocation/deallocation behavior by analyzing the program execution path. The cache miss behavior for two specified data structures, a linked list and a binary tree, was analyzed by tracking the memory allocation/deallocation sequence in synthetic programs [3]. However, in large and real programs, interactions and branch patterns are difficult to predict. These factors add complexity in extending this previous analysis.

In this paper, we avoid the detailed analysis of program execution path and use a mathematical model to extract the path pattern based on the observed paths. The primary contributions of this paper are:
1. The regularity of memory access patterns for procedures with pointer-based data structures is observed when using procedure call boundaries as the sampling unit.
2. A mathematical model, the Self-tuning Adaptive Predictor (SAP), is proposed to correlate both temporal and spatial locality with the program counter (PC). This model optimizes predictions of future memory addresses referenced by the program using a least-squares minimization technique [4].

2 Model Formulation

Consider a general example of the procedure call sequence in Figure 1. The memory access behavior of main( ) is complex as it jumps to different locations when different procedures are called. Its overall behavior depends on the behavior of all of the individual procedures. In the following analysis, a leaf procedure is a procedure that does not call other procedures. PC-correlated spatial locality occurs when the data address referenced by a load instruction at a PC (program counter) likely depends on memory addresses referenced by loads at nearby PCs. PC-correlated temporal locality occurs when the next memory address referenced by a load instruction at a certain PC is likely to depend on the previous memory addresses referenced by the same load instruction.

Fig. 1. Schematic of the calling procedure of a simple program.

This paper focuses on the memory access behavior produced by load instructions in leaf procedures using the procedure call as the fundamental sampling unit. The primary assumption is that, within some certain period, the behavior of memory accesses in a procedure depends on the history of both itself and nearby loads. This behavior can be represented as a linear system with constant but unknown parameters. At some point, the behavior changes which causes a consequent change in the specific parameter values. The goal of SAP is to detect such changes and automatically converge on the estimated parameter values as shown in Figure 2.

Consider the leaf procedure C in Figure 1. Suppose that there are r loads within procedure C with program counter (PC) values p_1, p_2, ..., p_r, respectively. Within the ith call of the procedure, the corresponding referenced addresses are denoted as A_{i,1}, A_{i,2}, ..., A_{i,r}. Within a certain range of consecutive calls, the behavior of memory accesses can be represented with the following equation, which takes both PC-correlated temporal and spatial localities into account:

\[ A_{n,m} = \sum_{i=1}^{l} a_{m,i} A_{n-i,m} + \sum_{i=0}^{k_1} a_{i,1} A_{n-i,1} + \cdots + \sum_{i=0}^{k_l} a_{i,l} A_{n-i,l} \]  

where 1 ≤ j ≤ (n-1), 0 ≤ k ≤ (n-1) (i=1,2,...,l), 0 ≤ l ≤ (m-1), and 1 ≤ m ≤ r.