Identifying Program Phase Behavior in Parallel Programs on Distributed Shared-memory Systems

Frederik Vandeputte*,[1], Koen De Bosschere*, Wim Heirman*

* *ELIS, Ghent University, Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium*

ABSTRACT

The execution of a typical program is not random, but follows certain patterns. In previous work several techniques have been proposed to extract and exploit these patterns for various purposes. Most of these studies were limited to single-threaded sequential programs. In this work, we analyze and extract the phase behavior of parallel programs and their network traffic on distibuted shared-memory systems using traffic-sensitive BBVs. The analysis of this phase behavior is divided into two parts, namely thread-level as well as program-level phase behavior. One of the main goals is to use this framework for latency reduction of network architectures with novel reconfigurable optical interconnection networks.

KEYWORDS: Program Phase Behavior, Network Traffic, Parallel programs, Splash-2, Shared-memory

1 Introduction

Most computer programs are not random, but consist of a number of repeating execution patterns, meaning that the execution of a given program can be divided into a number of key behavioral patterns, which are repeated during the execution of the program.

In the past, several techniques have been proposed to capture and exploit these repeating patterns. One popular technique is called Simpoint[Sher02], in which a number of program phases are extracted, by correlating execution behavior with executed program code. The execution of a program is divided into a number of fixed-length instruction intervals and for each instruction interval a *Basic Block Vector* (BBV) is collected. Each BBV basically is a histogram of how many times each basic block has been executed during some instruction interval. Once the BBVs are collected for a given program execution, they are grouped into a number clusters where each cluster contains similar BBVs. The resulting clusters – also called program phases – then reflect the different execution patterns that were captured during that execution of the given program.

In [Pere06], Perelman et al. extended the idea behind Simpoint for parallel programs. Using sampled BBVs, they first collect BBV information for each thread separately. Subsequentially, they cluster all BBVs of all threads of the parallel program to extract a number of phases, which can be shared across multiple threads. The local phases of the different threads are then synchronized to each other.

[1]This research was funded by Ghent University and by the Fund for Scientific Research-Flanders.

The remainder of this paper is not included as this paper is copyrighted material. If you wish to obtain an electronic version of this paper, please send an email to bib@elis.UGent.be with a request for publication P106.155.pdf.