PROSJEKTRAPPORT

Datamaskinarkitektur TDT4260

Trond Snekvik
trondesn@stud.ntnu.no

Christopher Benjamin Westlye
chriwes@stud.ntnu.no

Kaj Andreas Palm
kajpalm@gmail.com

Raymond Selvik
raymondd@stud.ntnu.no

Contents

1 (**2**

Abstract

A modern day computer has a CPU that have a tremendous processing power. However, the transmission rate on the RAM is still relatively low. This can be solved by adding a memory cache between the CPU and RAM. The performance of the cache is determent by how many memory hits the CPU does in the cache. If it does not hit anyone, it has to access it from the RAM anyway. If the the cache as a prefetcher that will determine what memorylocation that is going to be fetched next, the Memory misses will be tremendously decreased.

Chapter 1

Introduction

The performance of a modern CPU is much higher than the RAM. Most of the performance time will then be used to access the memory space of the RAM and load it into the CPU. This problem can be solved by adding a cache memory between the CPU and the RAM. Memory will then be access from the cache which has a much better performance. But the cache must also load memory from the RAM, and if the memory is wrong the CPU have to access the RAM anyway. In worst case the cache will have no effect.

The cache can be designed in many ways. It can for example be direct mapped which will have fastest hit times, but a fully associative will have the best trade off if there is many misses.But the cache design is deterministic and it does not know how the memory is mapped in a computer program. The performance can then be increased by adding a prefetcher. A prefetcher is a program that determines what the next memory call from the CPU will be, and load the memory location into the cache. The prefetcher can be designed by that fact that a computer program is containing many arrays and repetitive function which will have a known memory location relatively to each other.

The goal of this prosject is to make a prefetcher that has a algorithm that will decrease memory misses by the CPU. The soulution presented in this report is based on pattern regnonition implemented as a two dimensional vector structure...

Chapter 2

Background

Chapter 3

Prefetcher Description

The prefetcher has a pattern-seeking behaviour. It uses vectors to log observed fetch sequences, and ranks them based on how often the actual fetches correspond with the stored sequence. This is achieved by logging combinations of three requests, thus making a link between two fetches and what we expect to be referenced after these. Every time the two requests are recognized, the third one is prefetched. If this is correct behaviour, the score of that particular sequence is raised. The next fetch is then always assumed to be the third fetch in the sequence that has been correct the most times.

The access sequences that occur the least frequent are more prone to being removed in favour of new sequences. This is done by continually raising a threshold. Sequences that are recognized and proven to be correct have their value increased. If entries fall under the threshold,

Chapter 4

Methodology

Chapter 5

Results

Chapter 6

Discussion

Chapter 7

Conclusion