Chapter 10
Exploiting Spatial and Temporal Patterns in a High-Performance CPU

Goran Rakočević
Serbian Academy of Sciences and Arts, Serbia

Veljko Milutinović
Univeristy of Belgrade, Serbia

ABSTRACT

In modern computer systems, the effect known as the memory gap has become a serious bottleneck. It is becoming increasingly difficult to bridge this gap with traditional solutions, and much effort is put into developing new and more effective solutions to this problem. An earlier design, the Dual Data Cache (DDC), is a cache design that implies separation of data into two different cache subsystems so as to increase effectiveness of the cache. Data are separated accordingly to their predominant type of locality. The modified DDC, described here, introduces different internal organizations of the temporal and spatial parts, for better utilization of data characteristics. Conducted simulations show substantial improvements over traditional cache systems, with little increase in surface area and power consumption.

INTRODUCTION

With the development of the technology, the amount of comuputation that can be done in a unit of time within a microprocessor chip has grown dramatically. On the other hand, the speed at which data can be retervied from the main memory was not able to keep up. The result is a situation where microprocessor hardware remains underutilized, as it has to wait for inputs from the main memory, effectily making the memory a major system bottleneck. This phenomenon is known as the memory wall (McKee, 2004).

Memory wall has been a major target of research for decades (Wulf, 1995; McKee, 2004; Xie, 2013). The most effective way of allivating this problem has been the utilizations of cache mechanisams (Smith, 1987; Baer, 1988). A small,

DOI: 10.4018/978-1-4666-5784-7.ch010

very fast piece of memory is placed between the microprocessor chip and the main memory. This cache holds some of the data and respondes to requests form the CPU, while the main memory is accessed only when the data is not available in the cache. Cache memories soon became integrated with the microprocessor chips, enableing for very fast access times.

Further developments went in the direction of building data predictors – logic that could predict what data will be needed next, and than bring that data into the cache ahead of time (Lipasti & Shen, 1996). Unlike control flow (i.e. branch) predcitors, data predictors proved effective in only a small subset of cases, with relatively regular access patterns (Daly, 2013). An alternative solution was to offer pre-fetching primitieves, allowing the programmers to do this manulay, or thorugh compler support (Koufaty, 1998).

A number of approaches followed a different directon: to make the structure of the cache different and optimized to the data at hand. One such approach is presented here.

Problem Statement

Traditional cache memory architectures are based on the locality property of common memory reference patterns. This means that a part of the contents of the main memory is replicated in smaller and faster memories closer to processor. The processor can then access these data in the nearby fast cache, without suffering the long penalties of waiting for a main memory access.

Increasing the performance of a cache system is typically done by enlarging the cache. This has led to caches that consume an ever-growing part of modern microprocessor chips. However, bigger caches induce longer latencies, so making a cache larger, beyond a certain point, becomes counter-productive. At this point, a further increasing of the performance of the cache represents a difficult problem.

EXISTING SOLUTIONS

Most common solution to the above stated problem is the introduction of multiple level cache hierarchies, where the problem of speed is solved with fast and small lower level caches closer to the processor, while the problems of capacity and hit ratio are solved with slower and much larger higher level caches closer to the memory (Milutinovic, 1996). These high level caches occupy large portions of the chip, and induce high latencies in the system.

A number of attempts have been made towards developing mechanisms that improve performance of the cache by adding some logic to caching, rather than by increasing the cache size. Some of these mechanisms have focused on examining (physical) properties of cache memories and suggested solutions based on combing two or more cache memories of different characteristics on the same level (i.e. the Victim (Jouppi, 1990) and the Assist (Etsion, 2012) cache). These solutions mostly deal with increasing associativity of very fast direct-mapped cache memories, without incurring long latencies of highly associative caches. Others have looked at patterns of data accesses in order to build a cache system that suits the characteristics of data better (i.e. the Dual Data Cache, Split Temporal/ Spatial cache, etc.).

These solutions mostly deal with increasing hit ratio, or decreasing complexity (allow for smaller sized caches) without compromising hit ratio. A detailed taxonomy can be found in (Gašić, 2007).

The modified Dual Data Cache (modified DDC) falls in to the second category, as some of the concepts used in this chapter are taken from the Dual Data Cache (DDC), which was first introduced in a series of papers in the 1990' (González, 1995; Milutinovic, 1996; Milutinovic, 1996). However, the system also utilizes two cache memories of different characteristics on the same level, as the solutions in the first category.