Orthogonal Design Method for Optimal Cache Configuration

Hongsong Chen, Zhenzhou Ji, and Mingzeng Hu

Department of Computer Science and Technology
Harbin Institute of Technology
Harbin, 150001
chs68@pact518.hit.edu.cn

Abstract. A cache configuration is one of the most important factors to improve speed of transferring information between CPU(s) and memory. In order to find the optimum cache configurations that give high hit rate with small cache size, designers have to deal with many cache parameters such as number of sets for each cache, block size for each cache line, and so on. Beside the parameters, the designers have to find what the maximum hits of the cache under the consumption that the size is unlimited. Orthogonal method is apply modern statistical methods treating simultaneously several statistical variables. A purpose of this paper is to use orthogonal design method to search for the optimal cache configuration which produces high hit rate with small size. Using the orthogonal method for the cache configuration is successful because high hit rate with small size of cache is considered in the tests.

1 Introduction

Computer performance relies on many factors. One of them is cache efficiency to provide the necessary information to processor as soon as possible. In order to provide requested information to the processor, the cache has to search through all the cache space. If the size of the cache in fully set associative style is tremendous huge, searching time will be much longer than that in direct map. On the other hand, if lot of information needs to be kept in the cache, the set associative style mostly can store the data more than direct map with the same size. Processor companies have made central processing units with different cache configuration (Table 1).

There seem to be a little controversial when designers tried to design I-cache, D-cache. Some designers prefer to have more I-cache space than D-cache space, while others prefer to have more D-cache space than I-cache space. Another factor that designers need to consider is cache hierarchy. The cache can be designed to have one, two, or more levels. In general (might not be the best), the primary cache is split in to I-cache and D-cache, while the secondary cache is unified and able to dynamically allocate cache blocks to contain data or instruction depending on the program's requirements[1].

In the paper I proposes a technique using orthogonal method to find out which combinations give the near-optimal results based on hit ratio. The orthogonal method usually works well on large amount of data with a combination method to search near optimal results.

X. Zhou et al. (Eds.): APPT 2003, LNCS 2834, pp. 172–176, 2003.

Table 1. Example of on-chip memory in microprocessors

Processor	I-cache			D-cache		
	Size	assoc	line	Size	assoc	line
Intel Pentium	8-KB	2-way	8-word	8KB	2-way	8-word
DEC 21064 (Alpha)	8-KB	1-way	8-word	8-KB	1-way	8-word
Hitachi HARP-1	8-KB	1-way	8-word	16-KB	1-way	8-word
MIPS R4200	16-KB	1-way	8-word	8-KB	1-way	4-word
MIPS R4400	16-KB	1-way	8-word	16-KB	1-way	8-word
SuperSPARC	20-KB	5-way	16-word	16-KB	4-way	8-word
TeraSPARC	4-KB	1-way	8-word	4-KB	1-way	8-word

2 Methodology

2.1 Orthogonal Method

Experimental designs based on using orthogonal arrays change more than one variable to effectively determine the importance of several variables with a single test run. Orthogonal arrays were invented by Jacques Hadamar and first used for statistical experimentaldesign by Plackett and Burman[2].

The motivation is to minimise the size of an experiment, to control the amount of information required to be evaluated and to minimize the time and cost to execute an experiment. In this paper, I use it to select the optimum cache configuration.

2.2 Cache Architecture

Cache provides high speed data access for CPU to the memory by copying the portion of the program or data which will be used often into the cache line (cache block). How the caches are mapped to memories is important. And it also affects the performance of the computer[3]. Cache architecture are included with fully mapping, direct mapping and n-way set associative mapping.

2.3 Use Orthogonal Method to Design Cache

The main parameters used to construct the cache are number of set, block size, number of lines in one set and replacement policy. There are many kinds of combination, so we should analyze each factor and its level.

2.3.1 Assign Orthogonal Array

Because level one cache consists of instruction cache and data cache, I design that the parameters of level one instruction cache and data cache are equal. The level two cache is the uniform one. So we need design six factors. The number of columns

represents the levels of each kind of factors, every factor is parted into five levels. Factors and levels are showed in table 2.

Main cache's parameters are number of set (nset), block size (bsize), and number of lines in one set (assoc)[4]. In general, cache hierarchy has two levels and level one cache composes of instruction cache (I-cache) and data cache (D-cache).

Table 2. Fators and levels

Fctors	Level1	Level2	Level3	Level4	Level5
L1-nset	8	16	32	64	128
L1-bsize,byte	16	32	64	128	256
L1-assoc	1	2	4	8	16
L2-nset	16	32	64	128	256
L2-bsize,byte	128	256	512	1024	2048
L2-assoc	1	2	4	8	16

In my design, the parameters of level one data cache and level one instruction cache are equal. Therefore, there are six parameters of cache configuration.

The default replacement policy is LRU (least recently used), when we find the optimum cache configuration, I will change the replacement policy with FIFO(first in first out) and Random replacement policy, then compare them to find the best one.

2.3.2 Evaluation Function

Evaluation Function used in this paper is related to hit rate and cache size which are directly related to the performance of computers. If the number of hits is high, it means that the CPU seldom fetches the information from the slower storages. If the cache size is small, the seek time will be short.

This function will give the good, reasonable, and acceptable performance for the computer in the idea of speed and size of the caches because when the cache has the high hit rate and small cache size it will give the high evaluation value.

Evaluation function=100*il1_hit_rate + l1_weight /il1_size
+100*dl1_hit_rate + l1_weight /dl1_size
+100*ul2_hit_rate + l2_weight/ul2_size

il1_hit_rate: the rate of cache hits in level 1 instruction cache
dl1_hit_rate: the number of cache hits in level 1 data cache
ul2_hit_rate: the number of cache hits in level 2 uniform cache
l1_weight: the value of evaluating level 1 cache, designed with 10^6
l2_weight: the value of evaluating level 2 cache, designed with 10^6
il1_size: the size of level 1 instruction cache
dl1_size: the size of level 1 data cache ul2_size: the size of uniform 2 cache

The evaluation value of each cache consists of two parts. The first part is the rate of cache hits. The second part is the size of each cache. When the hit rate is high and the size of cache is small, the total evalution value is great.

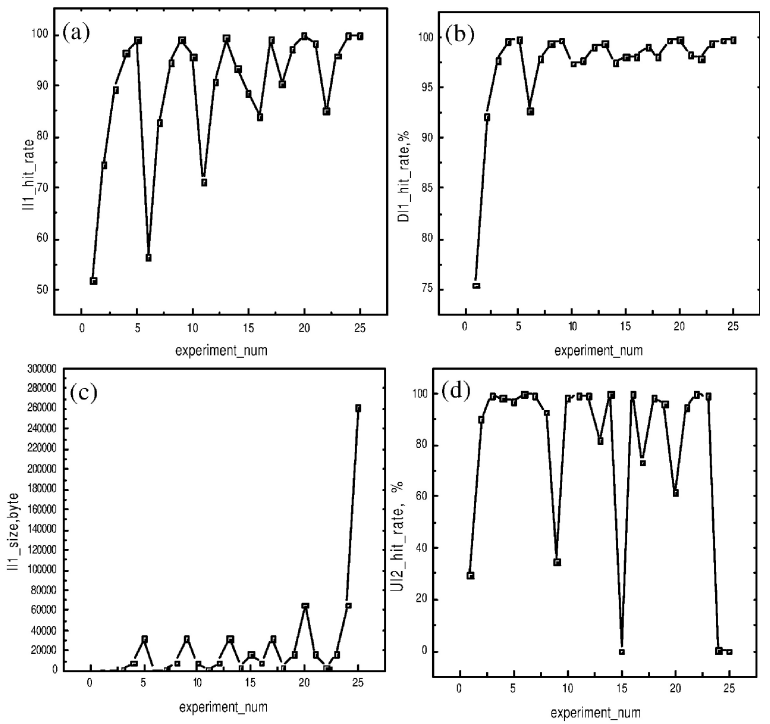
3 Testing Environment

Environment that I used to test includes hardware and simulation. The simulation will describe the parameters and benchmarks that I used for testing the idea.

On this paper, the test is based on Solaris 2.6 operating system. The simulator that I have used is simplescalar tool set[5]. The benchmark that I have used in this paper is test-math which is included in the Tool Set.

4 Result and Analysis

The result of each configuration is shown in Fig 1. Graphs show the change of evaluation function. As the rule of thumb, the bigger the cache size, the higher the hit rate.



(a). instruction level one cache hit rate (b). data level one cache hit rate
(c). instruction and data level one cache size (d). uniform level two cache hit rate

Fig 1. Results of Orthogonal experiment

As the Fig . 1 show, with the change of level one cache size, the instruction cache hit rate changes much more than that of level one data cache. When the parameters of level two cache changes, the level two cache hit rate changes much.

However, with the cache size starts to be bigger and bigger, the high hit rate is stable. Analyse the result data to determine the optimum levels of the factors. From Fig 1, we can selected the number of experiment 13, 14, 16 and 17 to find the fine cache parameters. Near optimum cache parameter and evaluation value List in table 3.

Table 3. Near optimum cache parameter and evaluation value

Experiment number	IL1_configure	DL1_configure	UL2_configure	Evaluation value
13	32:64:16	32:64:16	32:1024:1	372.03
14	32:128:1	32:128:1	64:2048:2	744.09
16	64:16:8	64:16:8	32:2048:4	342.89
17	64:32:16	64:32:16	64:128:8	347.63

Since on this paper, I give the importance of the evaluation function to the cache configuraton, we can select the cache parameter with quantitative analysis. The higher the cache hit rate, the smaller the cache size, the more the evaluation value is. See from table 3, we can select the best evaluation value set, the number 14, this cache configure- tion can get high hit rate with small cache size, it is fit to our design.

From the result above, I have selected the best cache parameters by much tests and theorise analyse. It is exact and accepted.

5 Conclusions

I bring forward a orthogonal design method to optimize the parameters of cache. The most important factors that affect the performance of the cache are hit rate and cache size. On this paper, the evaluation value is the function of these parameters. The evaluation value will increase when the number of hits increase and the cache size is small. From the results, I strongly believe that using the orthogonal design method to search for the best cache configuration is very effective.

References

1. Norman P. Jouppi and Steven J.E. Wilton.Tradeoffs in Two-Level On-Chip Caching. Compaq Computer Corporation, October, 1993
2. Owen, A.B. Orthogonal arrays for computer experiments, integration and visualization. Statist Sinica.1992,(2):439–452
3. David A. Patterson and John L. Hennessy. Computer Architecture, A Quan- titative Approach. Morgan Kaufmann Publishers Inc, 2002
4. Jeffrey B. Rothman and Alan Jay Smith.The Pool of Subsectors Cache Design.. Computer Science Division, University of California, Berkeley.1999
5. Doug Burger, and Todd M. Austin.The SimpleScalar Tool Set, Version 2.0. Computer Sciences Department, University of Wisconsin-Madison.1997