Curve fitting based shared cache partitioning scheme for energy saving

Kai Huang[1], Ke Wang[1a)], Xiaoxu Zhang[2], and Xiaolang Yan[1]

[1] *Institute of VLSI Design, Zhejiang University, Yuquan Campus,38 Zheda Road, Hangzhou, Zhejiang, 310027, China*

[2] *School of Information & Electronic Engineering, Zhejiang Gongshang University, 18 Xuezheng St., Hangzhou, Zhejiang, 310018, China*

a) wangke@vlsi.zju.edu.cn

Abstract: A linear and exponential (*sqrt2 rule*) combined curve fitting (CF) is proposed for low-energy shared cache partitioning. Considering cache's inherent characteristics between cache miss rate and its size, function with an exponent of (*1-sqrt2*) fits the region of non-linear high-utility cache size, while linear function fits both regions of linear high-utility and low-utility cache size. Using the fitted functions, we proposed a scheme with purely mathematical formulization of energy consumption, which helps fast and efficient shared cache partition. Experimental results show that CF based shared cache partitioning scheme achieves up to 34.5% energy savings compared with other traditional techniques. Moreover, our approach has a high prediction accuracy for the shared cache miss rate and the energy.

Keywords: Curve fitting, cache partition, energy efficient

Classification: Integrated circuits

References

[1] R. Reddy and P. Petrov: "Cache Partitioning for Energy-Efficient and Interference-Free Embedded Multitasking," ACM Trans. Embed. Comput. Syst. **9** (2010) 16 (DOI: 10.1145/1698772.1698774).

[2] M. K. Qureshi and Y. N. Patt: "Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches," Annual IEEE/ACM Int. Symp. on Microarchitecture (2006) 423 (DOI: 10.1109/MICRO.2006.49).

[3] H. Cook, *et al.*: "A hardware evaluation of cache partitioning to improve utilization and energy-efficiency while preserving responsiveness," ACM Int. Symp. On Computer Architecture (2013) 308 (DOI: 10.1145/2485922.2485949).

[4] G. Chen, *et al.*: "Cache partitioning and scheduling for energy optimization of real-time MPSoCs," IEEE 24th Int. Conf. on Application-Specific Systems, Architectures and Processors (2013) 35 (DOI: 10.1109/ASAP.2013.6567548).

©IEICE 2018
DOI: 10.1587/elex.15.20180886
Received September 21, 2018
Accepted October 10, 2018
Publicized October 30, 2018

[5] V. Venkatesan, *et al.*: "A 3-Level Cache Miss Model for a Nonvolatile Extension to Transcendent Memory," IEEE 6th Int. Conf. on Cloud Computing Tech. and Sci. (2015) 218 (DOI: 10.1109/CloudCom.2014.22).

[6] D. Zhan, *et al.*: "CLU: Co-Optimizing Locality and Utility in Thread-Aware Capacity Management for Shared Last Level Caches," IEEE Trans. Comput. **63** (2014) 1656 (DOI: 10.1109/TC.2012.277).

[7] A. Hartstein, *et al.*: "On the Nature of Cache Miss Behavior: Is It $\sqrt{2}$," Journal of Instruction-Level Parallelism **10** (2008) 1.

[8] GEM5: The gem5 Simulator System (2014) http://www.m5sim.org.

[9] C. J. Zhang, *et al.*: "A Highly Configurable Cache for Low Energy Embedded Systems," ACM Trans. Embed. Comput. Syst. **4** (2005) 363 (DOI: 10.1145/1067915.1067921).

[10] HP Labs: 'CACTI' (2015) http://www.hpl.hp.com/research/cacti.

1 Introduction

Reducing the energy consumption of the shared cache is essential in multi-application MPSoC platforms. Cache interference among multiple applications may increase the energy consumption of the memory subsystem and deteriorate its performance. There are many cache partitioning schemes [1, 2, 3] to avoid the interferences among different applications. However, few solutions carried out systematic analysis to minimize energy consumption, and cache miss rate minimization cannot always achieve energy savings. As for partitioning, integer linear programming (ILP) can achieve good results but consumes too much time [4]. Therefore, a systematic and effective cache partitioning scheme is necessary to derive the minimum energy consumption rapidly and efficiently.

In this letter, based on curve fitted energy formulation, we propose a cache partitioning scheme, which assigns cache ways to applications. Curve fitting is utilized in many works about caches, it is used for cache modeling [5] and management [6] and shows good feasibility and accuracy. Our work starts from analyzing the characteristic of cache miss rate, then two types of lines are chosen to fit the curve of cache miss rate according to the cache size. With the fitted curve function, cache partitioning problem for various goals is eventually transformed into a mathematical extremum problem. Both the static and dynamic power of the shared cache as well as the main memory are taken into account and the scheme finds a global optimal partitioning result with minimum energy consumption. All these works can be done at a very early stage of the design flow. Besides, the fitted curve is continuous, which can be used to solve cache partition problems of various granularities without sampling the data again.

2 Curve fitting based cache partitioning

The proposed scheme consists of three major steps, i.e., Memory Access Estimation (MAE), Curve Fitting (CF) and Cache Partition (CP). MAE is the first step. For each application, it outputs the number of cache misses under different configurations of way number, instruction number and memory access instruction number. CF is the second and key step which re-expresses the cache miss rate from discrete data obtained in MAE step to formulized functions. In order to fit the curve of miss rate, two issues must be considered.

First, the types of cache miss rate line are carefully selected. Cache miss rate resembles the *sqrt2 rule* when the cache size increases [7]. So it is reasonable to use the power function to fit the curve. Moreover, it is also obvious that the cache miss rate of applications shows linear relationship with the cache size, especially when the application is in cache saturate region [2]. Therefore, linear type curve is also suitable for fitting.

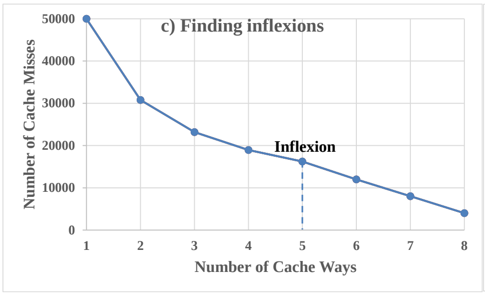
Another issue is to find the inflexions between sub-regions. Our scheme partitions a shared cache on way granularity. And the diversity of decrements is used to decide inflexions. The algorithm is shown in Fig. 1 (a) and an example is shown in Fig.1 (c). $N_m(n)$ denotes the miss number of memory references with *n* ways of the shared cache. Line 3 calculates the average decrement of $N_m(n)$. Line 4 presents the initial potential number of the inflexions, which is denoted as *INF*. As 1 and *W* cannot be the number of cache ways of the inflexions, *INF* equals to *W* minus 2. Line 7 to Line 14 try to find the inflexions, of which the amplitude of variation of the decrement of $N_m(n)$ exceed the threshold. And the threshold should be increased for the next iteration until the number of inflexions is no more than 2.

```
1 W = the way number of shared cache
2 K  = the ratio of diversity
3 Ave_dec = (N_m(1)-N_m(W))/W;
4 INF = W-2;
5 while (INF > 2){
6   i=0, j=0;
7   for(n=2; n<=W-1; n++){
8     dec0 = N_m(n-1) - N_m(n);
9     dec1 = N_m(n) - N_m(n+1);
10    thre = K×Ave_dec;
11    if(dec0 > thre)&&(dec1 < thre)){
12      inf1(i) = n; i++;}
13    else if((dec0<thre)&&(dec1>thre)){
14      inf2(j) = n;  j++;}}
15    INF = i+j;
16    increase K;}
```

a) Algorithm to find inflexions

```
1 (a1,b1,r1) = linear_fit($x_1,...,x_n,y_1,...,y_n$);
2 if(|r1|>=T){
3   Y = a1×X + b1;}
4 else{
5   for(i=1; i<=n; i++){
6     $x_i'' = \log_2(x_i); y_i'' = \log_2(y_i)$; }
7   (a2,b2,r2) = linear_fit($x_1'',...,x_n'',y_1'',...,y_n''$);
8   if((-0.7<a2<-0.3) && (|r2|>=T)){
9     $Y = 2^{b2} \times X^{a2}$;}
10 else{
11    (a2,b2,r2) = linear_fit($x_1,...,x_n,y_1,...,y_n$);
12    Y = a2×X + b2;}}
13  linear_fit($x_1,...,x_n,y_1,...,y_n$){
14    $a = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - \overline{x}^2}; b = \overline{y} - a\overline{x}; r = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\sqrt{(\overline{x^2} - \overline{x}^2)(\overline{y^2} - \overline{y}^2)}}$;
15    return a,b,r; }
```

b) Curve fitting for a sub-region



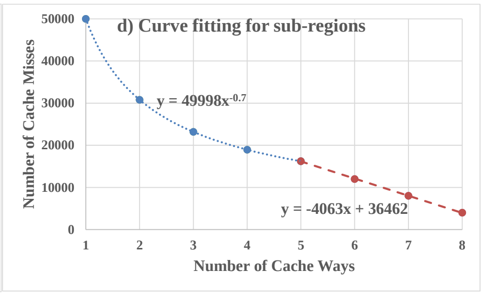


Fig. 1. Curve fitting process: (a) Algorithm to find inflexions, (b) Curve fitting for a sub-region, (c) Example of finding inflexions, and (d) Example of curve fitting.

Inflexions separate a curve of cache miss rate into three sub-regions at most as shown in our algorithm. And the types of curves are either linear or exponential with the power of about $(1-\sqrt{2})$ (-0.7 to -0.3 actually) [7]. So curve fitting can be done for each sub-region as shown in Fig. 1 (b) and an example is shown in Fig.

1 (d). The inputs are way numbers and the corresponding cache miss access numbers, denoted as (*x*, *y*) pairs in Line 1. The output is a miss rate function (MRF) of cache miss number with respect to the way number, i.e., $N_m = f(ways)$. From Line 1 to Line 3, if the correlation of the linear fitting is higher than the threshold, the MRF is obtained and it is a linear curve. From Line 5 to Line 9, the exponential fitting is performed. If the power is between -0.7 and -0.3 and the correlation is larger than the threshold, the MRF is derived and it is an exponential curve. Otherwise, linear curve is used as the MRF as shown in Line 11 and Line 12. The linear fitting algorithm is presented in Line 13 to Line 15. In our curve fitting algorithm, linear type with high correlation is given higher priority than exponential type with high correlation because exponential type has one extra transformation. If neither of the two types with high correlation can be used for fitting, linear type with lower correlation is used.

Once a fitted curve of cache miss rate for an application is obtained, we apply it to energy consumption function. The energy model in [9] is used, which has a comprehensive consideration for the energy consumption of a typical memory subsystem. The definitions of the variables are listed in Table I:

$$E_{mem} = E_{dyn} + E_{sta} \tag{1}$$

where

$$E_{dyn} = N_{hit} \times E_{hit} + N_m \times E_{miss} \tag{2}$$

$$E_{miss} = E_{offchip\_acc} + E_{cpu\_stall} + E_{line\_fill} \tag{3}$$

$$E_{sta} = E_{sta\_per\_cyc} \times Cycles \tag{4}$$

Table I. Definitions of variables in energy model

Variable	Definition	Variable	Definition
E_{mem}	Total energy,		Number of instruction:
E_{dyn}	Dynamic energy,	N_{acc}	Shared-cache access,
E_{sta}	Static energy	N_{inst}	All instructions,
	of memory subsystem	N_{ldst}	load/store memory access
	Energy of:		Energy of an access for:
$E_{offchip\_acc}$	Off-chip memory access, CPU stall for	E_{hit}	Hit on used cache,
E_{cpu_stall}	waiting off-chip data,	E_{miss}	Miss,
E_{line_fill}	Cache line fill	E_{tt_hit}	Hit when using total cache
	Static energy per cycle:		Number of
$E_{sta\_per\_cyc}$	used cache,	N_{hit}	Cache hit,
E_{tt_spc}	total cache	N_m	Cache Miss
S_c	Size of used cache	W	Ways of total cache
S_{tt}	Size of total cache	R_m	cache miss rate
$Cycles$	Execution cycles of an application		
C_{miss}	Cycle of a cache miss		
C_{hit}	Cycle of a cache hit		
$C_{ave\_}$	Average execution cycle of all instructions except memory access instructions		

It is assumed that the energy consumption is proportional to the activated size of the cache. Then we use memory access numbers and cache miss rate to derivate the equation:

$$E_{dyn} = N_{acc} \times (1 - R_m) \times E_{hit} + N_{acc} \times R_m \times E_{miss} \tag{5}$$

$$E_{sta} = E_{sta\_per\_cyc} \times (mem\_acc\_inst\_cycles + other\_inst\_cycles) \tag{6}$$

Combining the factors of cache miss rate (R_m) and cache size (S_c), we rewrite equation (5) and (6):

$$E_{dyn} = N_{acc} \times (S_c / S_{tt}) \times E_{tt\_hit} + N_{acc} \times R_m \times [E_{offchip\_acc} + E_{cpu\_stall} + E_{line\_fill} - (S_c / S_{tt}) \times E_{tt\_hit}] \quad (7)$$

$$E_{sta} = (S_c / S_{tt}) \times E_{tt\_spc} \times N_{acc} \times [C_{hit} + R_m \times (C_{miss} - C_{hit}) + C_{ave\_} \times (N_{inst} - N_{ldst}) / N_{acc}] \quad (8)$$

With the curve fitting result, we obtain:

$$S_c / S_{tt} = used\_ways / total\_ways = ways / W \quad (9)$$

$$R_m = N_m / N_{acc} = f(\text{ways}) / N_{acc} \quad (10)$$

From the equations (1) and (7) ~ (10) we can formulate the energy consumption of shared cache with different numbers of cache ways:

$$E_{mem}(ways) = A \times f(ways) \times (ways / W) + B \times (ways / W) + C \times f(ways) \quad (11)$$

A, B and C in Equation (11) are system or application related constant values which can be concluded from CPU specification, MAE and energy estimation in the proposed scheme flow, where

$$A = E_{tt\_spc} \times (C_{miss} - C_{hit}) - E_{tt\_hit}$$

$$B = E_{tt\_spc} \times N_{acc} \times (C_{hit} + C_{ave\_} \times (N_{inst} - N_{ldst}) / N_{acc}) + N_{acc} \times E_{tt\_hit}$$

$$C = E_{offchip\_acc} + E_{cpu\_stall} + E_{line\_fill}$$

In our scheme, the final step CP is to figure out the way numbers for each application and its corresponding energy consumption. Based on our curve fitting result, this is purely an arithmetic problem that can be solved by many available tools. A set of integer variables w_i are used to denote the number of ways assigned to the i^{th} application. And the problem statement is: for N applications and a W way shared cache, find a cache partitioning scheme that minimizes the energy consumption of the memory subsystem $\sum_1^N E_{mem}(w_i)$, with the constraints that $\sum_1^N w_i \le W$.

3 Experimental results

In order to evaluate the CF based cache partitioning scheme, we run integer applications in SPEC2006 on GEM5 [8] with ARM architecture in step MAE to prepare the discrete data for the next step CF. The simulator is configured with ARM Cortex-A8 architecture, 16KB L1 instruction and data cache per core, a 2MB 16-way associative shared L2 cache, and 512MB main memory. CACTI [10] is used to estimate the energy parameters of the shared cache.

We run each benchmark with at least 250M committed instructions. K and T in the algorithm of Fig. 1 are set to 1 and 0.7, respectively. Table II shows the average and the maximum error of the shared L2 cache miss rate between CF prediction and the simulated results. We compare L2 cache miss rates from 1 way to 16 ways for each benchmark. In Table II, most benchmarks show a very low average error which is less than 3%. Only 403.gcc and 483.xalancbmk are the exceptions which are a bit over 10%. Maximum prediction error follows the similar trend as average error, less than 15% error for all benchmarks except for the two mentioned (403 and 483) above. Benchmark 462.libquantum is also special, because it saturates as soon as one way is allocated and the miss rate keeps the same when the way number increases.

Table II. Error of CF predicted miss rate (%)

Benchmark	Ave. Err	Max Err	Benchmark	Ave. Err	Max Err
473.astar	0.004	0.035	462.libquantum	0	0
401.bzip2	1.684	14.345	429.mcf	2.929	10.344
403.gcc	10.770	39.279	471.omnetpp	0.698	4.811
445.gobmk	0.529	2.993	400.perlbench	2.229	10.986
464.h264ref	0.389	2.732	458.sjeng	0.013	0.057
456.hmmer	0.151	1.120	483.xalancbmk	13.878	34.607

We run multi-program on GEM5 multi-core processor to evaluate the energy savings with our cache partitioning scheme. Dual-, quad- and octo-core systems are evaluated, and their corresponding benchmark sets are grouped with 2, 4 and 8 benchmarks. All benchmark sets are executed under the condition that shared L2 cache is unpartitioned (MIX), even-partitioned (EVEN) and CF based partitioned (CFP). CFP results are also compared with the energy consumption calculated by traversing the whole exploration space (REF). All benchmarks run at least 250M instructions.

We compare the energy consumption with different partitioning schemes in Fig. 2. The energy of MIX are the basic reference normalized as 1 for comparison. As shown in Fig. 2, CFP and EVEN achieve much energy saving for all the benchmark sets, the reason is that all benchmarks have reached their saturation region with few cache ways and static energy dominates the total energy consumption. Static energy is proportional to the number of the cache ways used. MIX uses all the ways, but EVEN and CFP use much fewer. For dual-core benchmark sets, EVEN consumes 56.7% energy with respect to MIX, as the cache hit rate is high when each benchmark is assigned with half of the shared cache. CFP results show further reduction in energy consumption, i.e., 31.7% of MIX in average. The reason is that each benchmark uses fewer cache ways and many unused ways are shut down. The results are similar for quad-core and octo-core benchmark sets. But there is a significant shrink of the gap between EVEN and CFP, as fewer ways are turned off. For all benchmark sets, EVEN consumes only 46.1% energy of MIX on average, and CFP consume 65.5% of EVEN, i.e., 30.2% of MIX on average.

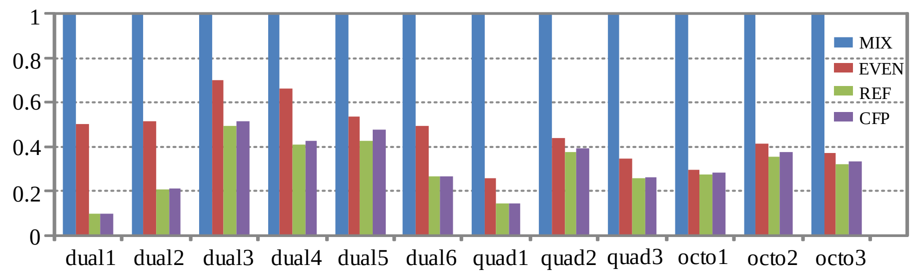


Fig. 2. Energy consumption of different partition scheme.

CFP always consumes less energy compared with MIX and EVEN, and it has nearly the same results as REF. As low as 3.3% predicting error are shown in average and 9.1% error at maximum. This means that the accuracy of CFP is high for predicting. Moreover, modulating the parameters in CF functions leads to better precision.

4 Conclusion

Based on the CF method, a rapid cache partitioning scheme is proposed with purely mathematical formulization of the energy cost, which helps fast and efficient shared cache partition. Experimental results demonstrate the energy efficiency of the proposed partitioning scheme. CF method achieves high energy prediction accuracy, with only 3.3% average error and 9.1% maximum error for final energy prediction. CF consumes 30.2% energy of MIX and 65.5% of EVEN on average.

Acknowledgments

This work was supported by National Key R&D Program of China (2018YFB0904900, 2018YFB0904902).