The Core Degree Based Tag Reduction on Chip Multiprocessor to Balance Energy Saving and Performance Overhead

Long Zheng, Mianxiong Dong, Hai Jin, Minyi Guo, Song Guo, Xuping Tu

The Core Degree Based Tag Reduction on Chip Multiprocessor to Balance Energy Saving and Performance Overhead

Long Zheng[1,2], Mianxiong Dong[2,3],*Hai Jin[1], Minyi Guo[4], Song Guo[2], and Xuping Tu[1]

[1] Huazhong University of Science and Technology, Wuhan, 430074, China
[2] University of Aizu, Aizu-Wakamatsu, 965-8580, Japan
[3] University of Waterloo, N2L, 3G1, Canada
[4] Shanghai Jiao Tong University, Shanghai, 200240, China
*hjin@hust.edu.cn

Abstract. Tag reduction is an approach to save energy of the cache system in a processor. Our previous work described that it can save more energy on a Chip Multiprocessor (CMP) than on a single-core processor. In this paper, we further investigate the problem on balancing energy saving and performance overhead when tag reduction is used for the low power Chip Multiprocessor (CMP). We first introduce the core degree concept which is defined as the number of cores that tag reduction can use for each thread. We then propose a core degree based tag approach that is to optimize the core degree such that the best balance of energy and performance can be achieved. In particular, as the basis for such optimization, the theoretical upper bounds of the energy savings and performance overhead are decided as function of the core degree. In our experiments, we use a 16-core CMP for example. In order to obtain the energy consumption and performance overhead with various core degrees, we construct an experimental environment, which is based on the Linux operating system. With the experimental environment, benchmarks of SPEC CPU2006 are used to evaluate our core degree based tag reduction. Finally, the experimental results show that the most desired balance of energy saving and performance overhead is achieved when core degree is set to 6.

1 Introduction

Over the last decade, manufactures competed to advance the performance of processors by raising the clock frequency. However, the dramatically increased power consumption and thermal problem caused by high clock frequency have ended this race. Now duplicating a number of cores in a chip, called Chip Multiprocessor (CMP) [1], as multi-core processor, is considered as an easier and more efficient way to enhance the performance of microprocessors. Multi-core architecture is quickly becoming the mainstream that can achieve higher computational capacity. Intel has acclaimed that more than 64 cores will be integrated into one

processor in the near future [2]. Nvidia has presented its many-core solution-GPGPU [3]. Moreover the multi-core architecture that is applied to embedded system and mobile computing has emerged recently [4].

It is well known that cache system is one of the most power consuming component in a processor, as it is implemented as SRAM to get a fast clock frequency. Taking two widely-used commercial processors as example, the cache system in Alpha 21164 uses up to 25% of the total energy consumed by the processor [5], and even higher in StrongARM-110 with the Instruction Cache (I-Cache) and Data Cache (D-Cache) of up to 26% and 17% [6], respectively. The less power dissipation means to relieve the thermal problem and prolong the battery lifetime in hand-held devices. Whereas, saving energy usually leads to performance degradation, so the energy and performance are always two major issues that should be well balanced in design and optimization of microprocessors.

This paper exploits tag reduction on Level 1 I-Cache (L1 I-Cache) to achieve energy and performance balance. We propose a method to extend tag reduction on CMP, meanwhile, introduce the core degree as the constraint to balance the energy and performance. The main idea of this paper is to distribute instruction page frames to several cores with the core degree constraint, such that the tag-reduction conflicts can be reduced or even avoided. This approach can improve the capacity of energy saving by tag reduction compared to the one on previous single-core processor. However, it causes the extra context switches as the overhead to assign instruction pages to multiple cores. The core degree constraint controls the overhead at all acceptable level by restricting of the number of cores that the instruction page frames are allowed to. By varying the values of core degree, we can find out the most appropriate value of core degree that achieves the best balance energy saving and performance overhead from our experiment.

It is difficult to get comprehensive data to measure the energy saving and performance overhead directly. Instead, we develop a model that can evaluate both energy saving and performance overhead of tag reduction quantitatively. Besides, since the results of tag reduction relies on the real distribution of instruction page frames, it is crucial to collect the distribution information of instruction page frames residing in physical memory, which we consider carefully in our experiments.

The remainder of this paper is structured as follows. Section 2 briefly presents the background of tag reduction and the related work. Section 3 propose the methodology that extends the tag reduction on CMP with energy saving and performance overhead, and presents a core degree based heuristic algorithm. Section 4 develops the model to evaluate the balance of energy and performance. Section 5 gives the experimental methods and results, as well as the analysis. Section 6 concludes our work and points out our findings.

2 Background and Related Work

In this section, we first review the basic concept of tag and tag reduction very briefly. Other related techniques on energy savings are also discussed. With a

Virtual-Indexed Physically-Tagged (VIPT) cache system of a processor, when the processor accesses L1 I-Cache, the 32-bit virtual address is divided into two parts. The most significant 20 bits constitute the Virtual Page Number (VPN) and the remainder is the page offset which is further divided into two 6-bit pars. The lower 6 bits are sent to processor directly, while the higher 6 bits are used as index to access the corresponding entry of the L1 I-Cache. Meanwhile, VPN is translated into Physical Page Number (PPN) through the TLB. Each entry of L1 I-Cache is composed by a tag and instructions. The tag stores the PPN of these instructions. Only if the tag in the entry is equal to the PPN translated by the TLB, called as L1 I-Cache hit, the instructions in the entry indexed by the most significant 6 bits of page offset are confirmed to be what the processor really needs to access. Otherwise, the L1 I-Cache miss leads to the secondary access to L2 I-Cache. In our paper, with the function of tag described above, shortening the length of tag can reduce the energy consumption of L1 I-Cache.

Tag reduction is a way to save energy consumed by L1 Cache and the Table Lookaside Buffer (TLB) in a processor [7] [8] [9] [10]. It has received much attention in the literature because it can save energy of caches significantly. Tag reduction has been applied to general or customizable embedded processor [7] [9], to the TLB [8], as well as to the processor using heterogeneously tagged caches [10]. Most research effects of tag reduction focus on its utilization on a single-core processor. Our previous work [11] is the first one that implements the tag reduction to the CMP by proposing three heuristic algorithms. However, it does not consider the performance issue, which motives us to further exploit tag reduction on CMP and to analyze the balance of energy saving and performance overhead.

3 Methodology

3.1 The Basic Idea of Tag Reduction on CMP

Tag reduction is an effective way to save energy of L1 cache system for two main reasons. One is that in order to achieve a fast clock, tag is implemented as the energy-consuming SRAM. The tag hardware is an array that includes several entries. All bits in a tag entry reside in a wordline. Each time when the tag entry is accessed, all bitlines are charged first and then discharged, resulting in a significant amount of energy consumption [7]. On the other hand, when Physical Address Extension (PAE) is used or not, a modern 32-bit processor can support up to 4GB or 64GB physical memory, respectively. However, it is a kind of waste to use 20-bit tag to distinguish the entries that have the same page offset, because OS does not always exhaust all the physical memory. Therefore, according to the organization of tag hardware, we can power off some unnecessary bits by reducing the length of a tag needed to save energy.

All cores in CMP shares Level 2 or Level 3 caches, but L1 caches are always private for each core. Therefore, each core only needs to distinguish the page frames that are assigned to itself. We first consider the ideal case that no tag-reduction conflicts will occur, when the tag reduction is applied to CMP. Under

such considerations, tag size B^i_{cmp} of each core i is

$$\begin{cases} B^i_{cmp} = \log_2 N_i \\ N = \sum_{i=0}^{w} N_i \end{cases} \quad (1)$$

where N_i is the number of page frames assigned to core i, N is the number of all page frames in physical memory and w is the number of the cores.

On the other hand, Equation (1) shows the best scenario is that all page frames are consecutively distributed in physical memory. When the page frames are not consecutive, the tag-reduction conflict exists [11]. Different from the tag reduction on a single-core processor, a well designed page frame assignment strategy on CMP can minimize or even avoid the tag-reduction conflicts. For example, we can separate the page frames, which would cause conflict, from each other by assigning them to different cores. The opportunities that allow tag reduction on CMP to decrease conflicts can achieve an even higher energy saving than the case on the single-core processor.

3.2 The Core Degree

For the notions of *thread* and *process*, OS does not always distinguish them all the time. Especially in process scheduling, OS treats threads and processes the same way. Since these differences do not affect the discussion of this paper, from now we do not distinguish a thread and a process explicitly, and we use the same word *thread* to represent them, except for some special cases when we need to distinguish them.

There are multiple cores in a CMP, therefore two or more threads can run parallel on different cores which is Thread Level Parallelism (TLP). This is how CMP gains performance improvement. Nevertheless, TLP on CMP would be much complicated. In order to decrease cache missing and achieve further optimization of each core, CMP with the support of operating system enables an affiliation mechanism that dedicates each thread running on a particular core, just like that each thread affiliates to a particular core. Since Windows NT 4.0 and Linux 2.4 supported Shared Memory Multiprocessor (SMP), implying that they also support CMP, the affiliation mechanism has been implemented. Because of affiliation mechanism, in the real CMP computer system, all I-Pages of a thread are assigned to a particular core, and the thread always runs on this core.

Based on the analysis in Section 3.1, tag reduction on CMP can exploit more opportunities to gain more energy saving than the one on a single-core processor by assigning page frames to different cores. In practice, tag reduction on CMP divides the whole page frames of each thread into several parts to decrease or avoid the tag-reduction conflicts. The core degree is thus introduced here to represent the number of cores that the page frames of a thread can be assigned to. The minimal core degree is 1 and the maximal is equal to the total number of cores. The core degree equal to 1 means that all the page frames of each thread

are assigned to one particular core. On the contrary, if core degree is equal to the number of cores, it means that page frames of each thread are assigned to all cores. The larger the value of core degree is, the more cores the page frames of each thread can be distributed to. In theory, a larger value of core degree provides opportunities to avoid tag-reduction conflicts, and at the same time, allows less number of page frames in average to be assigned to each core decreases. Both are beneficial to the energy savings.

Recall that tag reduction needs page frames of each thread to be divided into parts, we consider each part as non-parallel, as the data dependency between parts may exist. Each non-parallel part is designed to be treated as a thread by OS. The number of non-parallel parts depends on the decision of tag reduction on CMP. Because the concurrency of threads still remains, the non-parallel parts of different threads can run on different cores simultaneously. Besides, these non-parallel parts also follow the affiliation mechanism of OS, that is, each non-parallel part is always executed on the core that it is originally distributed by OS. Now, Equation (1) can be refined as

$$\begin{cases} B^i_{cmp} = \log_2 \sum_{j=1}^{p_i} N_{ij} \\ N = \sum_{i=1}^{d} \sum_{j=1}^{p_i} N_{ij}, d \leq \sigma \end{cases} \tag{2}$$

where N_{ij} is the number of page frames of the $j\text{-}th$ parts on core i, p_i is the number of parts on core i, d is the number of cores that have the parts and σ is the value of core degree.

Although tag reduction on CMP does not break the concurrency of threads, the non-parallel parts lead to extra thread switches. The thread switch takes system time, so the extra thread switches slow down the system. This is the performance overhead induced by tag reduction on CMP. Therefore, as the value of core degree goes up, the extra thread switches increases and then the performance of system goes down.

In a word, the core degree used by tag reduction on CMP affects both energy saving and the overhead of performance. When it grows, tag reduction can save more energy of CMP but causes more overhead of performance. The balance of energy saving and performance overhead is determined by core degree.

3.3 Heuristic Algorithm

In this section, we propose a heuristic algorithm to implement the method. The algorithm incorporates the affiliation mechanism and applies the tag reduction on CMP under a given core degree. As shown in the analysis above, the effect of tag reduction on CMP depends on the number of page frames of each core and the chance of tag-reduction conflicts. Since the tag-reduction conflict is always more crucial than the other, we consider it as the priority factor to design this heuristic algorithm. That is, we try to decrease and avoid the tag-reduction conflict as much as possible in this algorithm.

For loading instruction pages of Thread M, the heuristic algorithm can be described concisely as follows.

(1) Load a page of instructions that is part of Thread M into an I-Page frame with page frame number N.

(2) If the number of cores that the I-Page frames of Thread M have been assigned to is less than core degree, then go to step 3; otherwise go to step 4.

(3) In all cores, find out the one with the shortest reduced tag if the N-th I-Page frame is included; then go to Step 5.

(4) Within the cores that Thread M has already been assigned to, find out the one with the shortest reduced tag if the N-th I-Page frame is included.

(5) Dispatch the I-Page frame to the chosen core and make tag reduction.

(6) If more pages of instructions of Thread M need to be loaded into physical memory, return to Step 1; otherwise terminate.

4 Energy and Performance Analysis

In Section 3, we propose a tag reduction technique for a given core degree on CMP and discuss the corresponding energy saving and performance overhead issues conceptually. In this section, we develop an analytical model to quantitatively evaluate both energy consumption and performance overhead, as well as to study their balance.

4.1 Energy Consumption

Recall that accessing a tag once undergoes a cycle of charging and discharging as described in Section 3.1. We assume the energy consumption of one such cycle for one bit tag to be e Joules. When programs run on a processor, some instructions are jumped over and, on the contrary, some are executed repeatedly. However, the instructions are executed by a processor like an instruction flow from view of the processor. Each instruction fetching from cache and memory hierarchy needs to access L1 I-Cache tag. Because the execution of different instructions takes different time, the number of instructions executed in a processor time slice may be not equal.

For a CMP with w cores, running t threads, the energy consumption E on the L1 I-Cache tag in a period with s time slices can be calculated as follows.

$$E = e \cdot \sum_{i=1}^{s} \sum_{j=1}^{w} \sum_{n=1}^{a_{ij}} TagR(\bigcup_{m=1}^{t} AddEx(I_{ij}(m,n))), \tag{3}$$

subject to $\forall i, 1 \leq i \leq s$, $\forall m, 1 \leq m \leq t$, and $\forall n, 1 \leq n \leq a_{ij}$

$$\sigma \geq w - \sum_{j=1}^{w} \phi(I_{ij}(m,n)). \tag{4}$$

The variables and functions in (3) and (4) are detailed as follows. $\forall i, 1 \leq i \leq s$ and $\forall m, 1 \leq m \leq t$, we use $I_{ij}(m,n)$ to denote a set that represents thread m's

instruction flow till the n-th instruction executed by the j-th core during the i-th time slice. Each element records a particular instruction's physical address and execution time. In the instruction flow, the element of $I_{ij}(m,n)$ is a two-dimensional array, just like (A,T) in which A is the physical address and T is the executed time. The n-th instruction can be expressed as (A_n,T_n). Therefore the thread m's instruction flow till the n-th instruction can be expressed exactly as

$$\{(A,T)|(A,T)\in I'_{ij}(m),T\leq T_n\}, \tag{5}$$

where $I'_{ij}(m)$ represents thread m's instruction flow executed by the j-th core during the i-th time slice.

Function $AddEx$ extracts the A of each (A,T) of set $I_{ij}(m)$ to compose a new physical address set. Function $TagR$ reduces the length of tag with the physical addess set $AddEx(I_{ij}(m))$. Furthermore, a_{ij} is the total number of instructions executed by the j-th core, during the i-th time slice, i.e.

$$a_{ij}=num(\bigcup_{m=1}^{t}I'_{ij}(m)) \tag{6}$$

Finally, Equation (4) guarantees that instructions of each thread can be only assigned to at most σ number of cores, due to the core degree constraint. It is formulated by the function $\phi(S)$ which returns 1 if set S is an empty set or 0 otherwise.

We further define the average number of bits that a tag requires after tag reduction for the j-th core to execute all the instructions in the i-th time slide, $\overline{B_{ij}}$, as follows.

$$\overline{B_{ij}}=\frac{1}{a_{ij}}\cdot\sum_{n=1}^{a_{ij}}TagR(\bigcup_{m=1}^{t}AddEx(I_{ij}(m,n))) \tag{7}$$

Equation (3) can thus be rewritten as

$$E=e\cdot\sum_{i=1}^{s}\sum_{j=1}^{w}(\overline{B_{ij}}\cdot a_{ij}) \tag{8}$$

Let N_i be the total number of instructions executed by all cores in the i-th time slice, i.e.,

$$N_i=\sum_{j=1}^{w}a_{ij} \tag{9}$$

and f_{ij} be the ratio of the number of instructions assigned to the j-th core over the total number of instructions to all cores in the i-th time slice, i.e.,

$$f_{ij}=\frac{a_{ij}}{\sum_{j=1}^{w}a_{ij}}=\frac{a_{ij}}{N_i} \tag{10}$$

Equation (3) can be eventually expressed as

$$E = e \cdot \sum_{i=1}^{s} \sum_{j=1}^{w} (\overline{B_{ij}} \cdot f_{ij} \cdot N_i) \tag{11}$$

If we define

$$N_{max} \equiv \max_{1 \le i \le s} (N_i) \tag{12}$$

the upper bound of energy consumption by tag reduction $\mathcal{E}$ can be derived as

$$\begin{cases} E \le \mathcal{E} = e \cdot N_{max} \cdot \sum\limits_{i=1}^{s} \sum\limits_{j=1}^{w} (\overline{B_{ij}} \cdot f_{ij}) \\ \sigma \ge w - \sum\limits_{j=1}^{w} \phi(I_{ij}(m,n)) \end{cases} \tag{13}$$

We have obtained the upper bound of energy consumption with tag reduction under the core degree constraints as formulated in (13). The upper bound of energy consumption can be calculated if $\overline{B_{ij}}$ and f_{ij} are known, since both e and N_{max} are constant numbers.

4.2 Performance Overheads

To formulate the discussion in Section 3.2, the performance overhead is defined as the number of the additional thread switches induced by tag reduction. We first show the total number of switches, ST, on a tag reduction enabled CMP as

$$ST = SP + SR - \varepsilon, \varepsilon \ll SP, SR \tag{14}$$

where SP is the number of thread switches when tag reduction is not used, SR is the number of thread switches caused by the additional non-parallel parts, and ε is the overlap between SP and SR. Generally, the overlap rarely happens, only in some very special cases, for example, when a thread switch caused by additional non-parallel parts should have occurred because of the expiration of time slice of OS. So ε is far less than SP or SR. The performance overhead caused by tag reduction on CMP T is

$$T = SR - \varepsilon \tag{15}$$

Similar to (3), for a CMP with w cores and t running threads, SR in a period with s time slices can be calculated as

$$SR = \sum_{m=1}^{t} \sum_{i=1}^{s} \frac{F_{im} - 1}{P_{im}} \sum_{j=1}^{w} a_{ij} \tag{16}$$

subject to $\forall i, 1 \le i \le s$ and $\forall m, 1 \le m \le t$

$$F_{im} \le \sigma \tag{17}$$

In (16) P_{im} is the number of I-Pages of the m-th thread during the i-th time slice, F_{im} is the number of non-parallel threads that are generated by the m-th thread during the i-th time slice; and the definition of a_{ij} is the same as in (6). Due to the core-degree constraint, F_{im} must not exceed σ, which is described by (17).

Similarly, using N_i defined in (9), we can simplify (16) as

$$SR = \sum_{m=1}^{t} \sum_{i=1}^{s} \frac{N_i}{P_{im}} (F_{im} - 1) \tag{18}$$

So, T can be expressed as

$$T = \sum_{m=1}^{t} \sum_{i=1}^{s} (N_i \cdot (F_{im} - 1)/P_{im}) - \varepsilon \tag{19}$$

The upper bound of the performance overhead $\mathcal{T}$ caused by extra non-parallel parts when tag reduction applied to CMP can eventually be derived as

$$\begin{cases} T \leq \mathcal{T} = N_{max} \cdot \sum\limits_{m=1}^{t} \sum\limits_{i=1}^{s} ((F_{im} - 1)/P_{im}) \\ F_{im} \leq \sigma \end{cases} \tag{20}$$

We notice that $\mathcal{T}$ is only related to F_{im} and P_{im}, since N_{max} and SP are constants for each parallel thread. When the tag reduction is not used, that is, F_{im} is always equal to 1, $\mathcal{T}$ is equal to 0, as expected.

4.3 Balance of Energy and Performance

In order to facilitate the comparison of the energy consumption and performance overhead under various core degrees for a group of applications, we introduce the normalized metrics θ and ω as follows.

$$\theta_n^\sigma = \mathcal{E}_n^\sigma / \mathcal{E}_n^{\sigma^{min}}, \theta_n^\sigma \in (0, 1] \tag{21}$$

and

$$\omega_n^\sigma = \mathcal{T}_n^\sigma / \mathcal{T}_n^{\sigma^{max}}, \omega_n^\sigma \in (0, 1] \tag{22}$$

in which θ_n^σ is the ratio of energy consumption of the n-th application when core degree is equal to σ relative to the one when core degree is σ^{min}, and ω_n^σ is the ratio of performance overhead of the n-th application when core degree is set to σ relative to the one when core degree is σ^{max}. As we apply the tag reduction on CMP, the minimal value of core degree σ^{min} is 2. The maximal value of core degree σ^{max} is equal to the number of cores. Similarly, $\mathcal{E}_n^\sigma$ and $\mathcal{T}_n^\sigma$ represent the energy consumption and performance overhead of the n-th application, respectively, when core degree is equal to σ. Based on our analysis, as core degree σ increases, θ_n^σ goes down from 1, and ω_n^σ goes up to 1.

Theoretically, the energy consumption always decreases and the performance overhead always increases as core degree varies from 2 to the number of cores. It is meaningless that compare the energy consumption and performance overhead directly. Instead, we could compare the trends of energy consumption with the one of performance overhead of different applications as core degree varies to find out the balance of the energy and performance of tag reduction on CMP. So the trend of energy consumption and performance overhead, can be calculated as

$$\eta_n(\sigma) = -\frac{d\theta_n}{d\sigma}, \sigma \in (\sigma^{min}, +\infty) \tag{23}$$

and

$$\gamma_n(\sigma) = \frac{d\omega_n}{d\sigma}, \sigma \in (\sigma^{min}, +\infty) \tag{24}$$

where η and γ are the trends of energy consumption and performance overhead of the n-th application, respectively. Equation (23) makes sure that the trend of energy consumption η is positive, since energy consumption usually goes down as core degree increases so that the value of $\frac{d\theta_n}{d\sigma}$ is negative.

So far, we can use the trends of energy consumption and performance overhead to find out the appropriate value of core degree. In microprocessor design, it is demanded to consume less energy with low performance overhead. However, with the tag reduction on CMP, the energy consumption always goes down and the performance overhead always goes up, so that with a particular value of core degree, the bigger the value of η_n is and the less the value of γ_n is, the better the core degree is. Therefore, the appropriate value of core degree of each application can be determined as follows:

$$\sigma_n = \begin{cases} \arg\min_{\sigma}(|\gamma_n(\sigma) - \eta_n(\sigma)|) & : \text{ Case 1} \\ \arg\min_{\sigma}(\gamma_n(\sigma) - \eta_n(\sigma)) & : \text{ Case 2} \\ any \in [\sigma^{min}, \sigma^{max}] & : \text{ Case 3} \\ \arg\min_{\sigma}(\gamma_n(\sigma) - \eta_n(\sigma)) & : \quad else \end{cases} \tag{25}$$

where Case 1, 2, 3 can be expressed as follows.
Case 1:

$$\eta_n(\sigma^{min} + \Delta) > \gamma_n(\sigma^{min} + \Delta) \text{ and}$$
$$\exists \sigma' \in (\sigma^{min}, +\infty), \eta_n(\sigma') = \gamma_n(\sigma')$$

Case 2:

$$\eta_n(\sigma^{min} + \Delta) < \gamma_n(\sigma^{min} + \Delta) \text{ and}$$
$$\exists m' \in (\sigma^{min}, +\infty), \eta_n(m') = \gamma_m(m')$$

Case 3:

$$\eta_n(\sigma) \equiv \gamma_n(\sigma) + C, C \text{ is a constant number}$$

In 25, σ_n is the appropriate value of core degree of the n-th application. There are 3 situations when we decide the best value of core degree described in the

equation above. If we can find a common value of σ_n for sorts of applications, we can know the best value of core degree that achieves the balance of the energy consumption and performance overhead of tag reduction on CMP.

5 Experiments

In our experiments, we use 22 benchmarks from SPEC CPU2006 to find out the appropriate value of core degree, as the benchmarks from SPEC CPU2006 represent sorts of classic applications. First, we construct an experiment platform to evaluate 22 benchmarks and collect necessary data; then we calculate the energy consumption, performance overhead and their trend as described in Section 4; finally, the appropriate value of core degree can be got through the ones of 22 benchmarks.

5.1 Experiment Setup

We choose Linux 2.6.11 32bit without PAE as our experiment platform, because Linux is open source, so that it is easy for us to modify the kernel and add some modules to collect data needed. Three copies of each benchmark are run concurrently every time to create a multi-process environment. After modifying the kernel and adding extra module, we can collect the experimental data from these 3 copies of each benchmark.

As analyzed above, the data needed to calculate energy and performance only concerns the physical memory information. There are no differences in physical memory management of Linux between CMPs with different number of cores. Therefore, with the data we collected, we construct a 16-core CMP environment to calculate the energy consumption and performance overhead that are basis of computation of their balance. The experiment environment is listed as that Processor Model is Intel Core2 6400@2.13GHz and physical memory capacity is 1GB.

The names of 22 benchmarks evaluated are astar, bwaves, bzip2, cactusADM, calculix, dealII, gcc, GemsFDTD, gromacs, h264ref, hmmer, lbm, libquantum, namd, perlbench, povray, sjeng, soplex, sphinx, tonto, Xalan, and zeusmp. The benchmarks consist of integer and float point computation. Since in our experiments, we apply the tag reduction on a 16-core CMP, the range of core degree is from 2 to 16, i.e. σ_{min} and σ_{max} are 2 and 16, respectively. We evaluate each of 22 benchmarks with different values of core degree from 2 to 16 with interval by 2.

5.2 Experiment Result and Analysis

Based on the analysis in Section 4, with the data collected from our experiment platform, we can calculate θ_n^σ and ω_n^σ that are energy consumption ratio and performance overhead ratio of each benchmark when core degree varies from 2

Table 1. The Discrete Results of Energy Consumption as Core Degree Varies (θ_n^σ).

CD	4	6	8	10	12	14	16	CD	4	6	8	10	12	14	16
astar	0.74	0.54	0.43	0.38	0.35	0.33	0.29	bwaves	0.67	0.48	0.42	0.39	0.35	0.37	0.32
bzip2	0.59	0.40	0.29	0.29	0.29	0.30	0.32	cactusADM	0.72	0.63	0.53	0.53	0.49	0.47	0.45
calculix	0.66	0.57	0.50	0.45	0.43	0.43	0.41	dealII	0.84	0.66	0.58	0.53	0.50	0.49	0.43
gcc	0.67	0.56	0.50	0.51	0.52	0.46	0.46	GemsFDTD	0.58	0.47	0.39	0.34	0.31	0.29	0.29
gromacs	0.87	0.75	0.67	0.60	0.56	0.49	0.48	h264ref	0.74	0.54	0.52	0.45	0.44	0.40	0.38
hmmer	0.71	0.60	0.53	0.50	0.46	0.42	0.40	lbm	0.70	0.49	0.37	0.33	0.28	0.23	0.27
libquantum	0.74	0.59	0.52	0.44	0.37	0.34	0.29	namd	0.63	0.51	0.41	0.40	0.34	0.32	0.30
perlbench	0.87	0.76	0.74	0.72	0.62	0.62	0.62	povray	0.73	0.68	0.60	0.54	0.47	0.45	0.44
sjeng	0.75	0.58	0.52	0.46	0.41	0.38	0.35	soplex	0.73	0.64	0.55	0.47	0.44	0.41	0.37
sphinx	0.67	0.55	0.48	0.45	0.41	0.37	0.34	tonto	0.83	0.71	0.67	0.60	0.58	0.54	0.53
Xalan	0.74	0.63	0.55	0.51	0.50	0.44	0.43	zeusmp	0.79	0.59	0.50	0.42	0.36	0.38	0.38

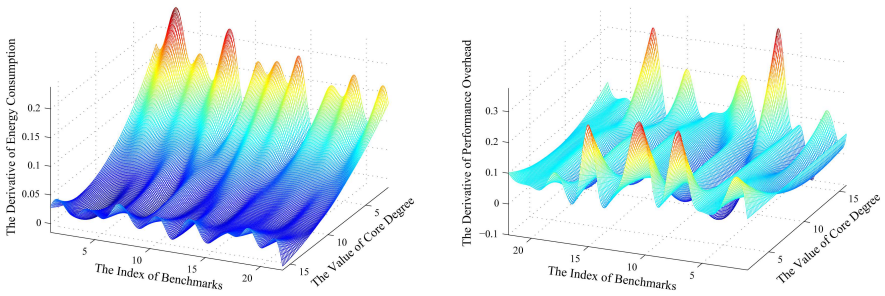
Table 2. The Discrete Results of Performance Overhead as Core Degree Varies (ω_n^σ).

CD	2	4	6	8	10	12	14	CD	2	4	6	8	10	12	14
astar	0.07	0.20	0.33	0.47	0.60	0.73	0.87	bwaves	0.26	0.62	0.75	0.87	0.88	0.87	0.97
bzip2	0.08	0.25	0.42	0.57	0.56	0.68	0.83	cactusADM	0.52	0.83	0.96	0.56	1.38	1.25	0.90
calculix	0.07	0.20	0.33	0.47	0.60	0.73	0.87	dealII	0.09	0.26	0.42	0.58	0.67	0.73	0.87
gcc	0.23	0.66	0.68	0.50	0.71	0.10	0.54	GemsFDTD	0.37	0.75	0.86	0.89	0.96	0.97	1.00
gromacs	0.07	0.20	0.33	0.47	0.60	0.73	0.87	h264ref	0.22	0.65	0.67	0.70	0.72	0.75	0.36
hmmer	0.29	0.88	0.90	0.92	0.94	0.96	0.7	lbm	0.17	0.43	0.61	0.65	0.76	0.79	0.97
libquantum	0.07	0.20	0.33	0.47	0.60	0.73	0.87	namd	0.07	0.20	0.33	0.47	0.60	0.73	0.87
perlbench	0.11	0.34	0.56	0.79	0.74	0.52	0.47	povray	0.07	0.20	0.34	0.47	0.60	0.73	0.87
sjeng	0.07	0.20	0.33	0.47	0.60	0.73	0.87	soplex	0.04	0.09	0.13	0.16	0.20	0.24	0.26
sphinx	0.07	0.20	0.33	0.47	0.60	0.73	0.87	tonto	0.07	0.20	0.34	0.47	0.60	0.73	0.87
Xalan	0.07	0.20	0.34	0.48	0.60	0.72	0.86	zeusmp	0.09	0.14	0.39	0.50	0.55	0.58	0.84

to 16 with interval by 2. Different values of n mean different benchmark. Because we evaluate 22 benchmarks, the n is from 1 to 22.

Table 1 illustrates the energy consumption ratio of each benchmark when core degree is equal to 2, 4, till 16. CD in the table is short for core degree. The values of ratio are distributed discretely by the discrete values of core degree, because the original experimental results are discrete. With this table, we can find out that the energy of consumption of all the benchmarks decreases when the value of core degree increases. Because the value of energy consumption ratio of all benchmarks is 1 when core degree is 2 as the definition of normalization in (21), we omit these values in the table. Table 1 and the theoretical analysis in Section 4.1 fit neatly.

Similarly in Table 2, it shows the performance overhead ratio of each benchmark. With the definition of normalization in (22), the performance overhead ratio is 1, when core degree is 16. However, we notice that the maximum of performance overhead ratio of some benchmarks exceeds 1. The performance overhead ratio for most of 22 benchmarks are monotonic increment along the



(a) The results of derivative of energy consumption as core degree varies.

(b) The results of derivative of performance overhead as core degree varies.

Fig. 1. The discrete results of derivative of energy consumption and performance overhead ratio

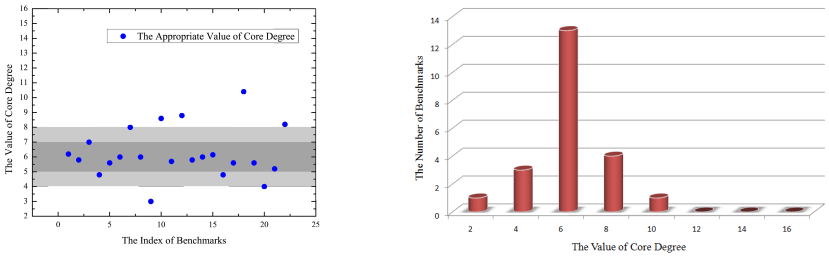
varying of core degree from 2 to 16. However, some benchmarks behave more complicatedly than monotonically. The results of performance overhead ratio from experiments are a bit more complex than the theory in Section 4.2.

With the results above, we should follow the analysis in Section 4.3 to get the trends of energy consumption and performance overhead of each benchmark, η_n and γ_n, respectively. This is basis of determining the appropriate value of core degree. In order to get η_n and γ_n, we fit the energy consumption ratio function and performance overhead ratio function with the their discrete results with help of Matlab. With the functions fitted, we can get the expressions of derivative of these functions, so that the trends are available for calculation.

In Fig. 1, we index each benchmarks from 1 to 22 alphabetically, instead of using its name for convenience. Fig. 1 shows the results of derivative of energy consumption and performance overhead ratio, respectively, when core degree is set to 2, 4 till 16. The x-axis is the value of σ, the y-axis is the index of benchmarks. The z-axis in Fig. 1(a) is the results of derivative of energy consumption, the one in Fig. 1(b) is the results of derivative of performance overhead.

In Fig. 1(a), the results of derivative of energy consumption trends to 0 along the σ varies, which implies that the effect of energy saving by tag reduction on CMP is weaker and weaker when core degree varies incrementally. We find out that the results of derivative of energy consumption vary similarly among all benchmarks. When core degree is greater than 12, part of benchmarks achieve their low peaks, and others then still remain monotonic, which means after Core Degree is 12, part of benchmarks can save more energy, the others are otherwise.

The results of derivative of the performance overhead are shown in Fig. 1(b). We notice that the trends of most of benchmarks remain stable; the ones of others are otherwise. Besides, the negative values appear several times in this figure, which means the performance overhead decreases when core degree is equal to some particular values. It is obvious that all benchmarks can be divided into two parts: Part A and Part B. The derivative of performance overhead ratio



(a) (b)

Fig. 2. The appropriate value of core degree

of benchmarks in Part A stays almost the same; on the contrary the one in Part B changes dramatically as the value of core degree varies.

With the results of derivative of the fitted energy consumption and performance overhead ratio of each benchmark, we can get the appropriate value of core degree of each benchmark, i.e. σ_n, according to (25) in Section 4.3. The distribution of σ_n is illustrated in Fig. 2(a). Each point in this figure shows a particular benchmark's appropriate value of core degree. The x-value of a point is the index of a benchmark, and the y-value of the point is the appropriate value of core degree of this benchmark. There are two districts in this figure. One is the filled with light grey and the other is with dark grey. The light grey district is a district around core degree 6 with radius 2; and the dark grey district is a district around core degree 6 with radius 1. The appropriate values of core degree of 22 benchmarks are distributed between 3 and 10. Most of benchmarks are in light grey district, especially in dark grey district. Fig. 2(b) offers the distribution with the number of benchmarks more clearly. In this figure, the x-axis is the value of core degree, the y-axis is the number of benchmarks. Taking the cylinder at core degree 6 for example, this cylinder represents that the appropriate values of 13 benchmarks is in the interval of (5,7]. Therefore, we can see that all appropriate values of core degree of benchmarks are from 2 to 10, with a peak at core degree 6 that is much higher than ones at other core degree. As the benchmarks from SPEC CPU2006 represent sorts of typical applications, from the results in Fig. 2, we can find out when core degree is 6, it is appropriate for applications to balance energy consumption and performance overhead.

6 Conclusion

In this paper, we first review the fundamentals of tag reduction on CMP that can save energy consumption. Then we introduce the core degree concept, meanwhile propose the core degree based approach to tag reduction to balance the energy consumption and performance overhead. Furthermore the analysis of balance of energy and performance leads to finding out an appropriate core degree. In the

experiment, we apply our core degree based tag reduction to the 16-core CMP, and then use 22 benchmarks of SPEC CPU2006 to evaluate our approach, finally find out the optimal value of core degree. The experimental results show that when the core degree is 6, we can get the most desired balance of energy saving and performance overhead.

7 Acknowledgements

This work is supported by National 973 Basic Research Program of China under grant No. 2007CB310900, the National High-Tech Research and Development Plan of China (863 Plan) under Grant Nos. 2008AA01Z106, and Fellowships of the Japan Society for the Promotion of Science for Young Scientists Program, Excellent Young Researcher Overseas Visit Program.

References

1. Spracklen, L., Abraham, S.: Chip multithreading: Opportunities and challenges. In: Proceedings of the 11th International Symposium on High-Performance Computer Architecture, IEEE Computer Society Washington, DC, USA (2005) 248–252
2. Held, J., Bautista, J., Koehl, S.: From a few cores to many: A tera-scale computing research overview. Research at Intel white paper (2006)
3. Lindholm, E., Nickolls, J., Oberman, S., Montrym, J.: NVIDIA Tesla: A unified graphics and computing architecture. IEEE Micro (2008) 39–55
4. Monchiero, M., Palermo, G., Silvano, C., Villa, O.: Efficient synchronization for embedded on-chip multiprocessors. **14**(10) (2006) 1049–1062
5. Edmondson, J., Rubinfeld, P., Bannon, P., Benschneider, B., Bernstein, D., Castelino, R., Cooper, E., Dever, D., Donchin, D., Fischer, T., et al.: Internal organization of the Alpha 21164, a 300-MHz 64-bit quad-issue CMOS RISC microprocessor. Digital Technical Journal **7**(1) (1995) 119–135
6. Montanaro, J., Witek, R., Anne, K., Black, A., Cooper, E., Dobberpuhl, D., Donahue, P., Eno, J., Hoeppner, W., Kruckemyer, D., et al.: A 160-mhz, 32-b, 0.5-w CMOS RISC microprocessor. IEEE Journal of Solid-State Circuits **31**(11) (1996) 1703–1714
7. Petrov, P., Orailoglu, A.: Dynamic tag reduction for low-power caches in embedded systems with virtual memory. International Journal of Parallel Programming **35**(2) (2007) 157–177
8. Petrov, P., Orailoglu, A.: Virtual page tag reduction for low-power TLBs. In: Computer Design, 2003. Proceedings. 21st International Conference on. (2003) 371–374
9. Zhou, X., Petrov, P.: Heterogeneously Tagged Caches for Low-Power Embedded Systems with Virtual Memory Support. ACM transactions on design automation of electronic systems **13**(2) (2008) 32
10. Petrov, P., Orailoglu, A.: Tag compression for low power in dynamically customizable embedded processors. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **23**(7) (2004) 1031–1047
11. Zheng, L., Dong, M., Guo, S., Guo, M., Li, L.: I-Cache Tag Reduction for Low Power Chip Multiprocessor. In: 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications. (Chendu and Jiuzhai Valley, China, Auguest 10-12, 2009) 196–202