TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 10/11 pp468–483
DOI: 10.26599/TST.2018.9020104
Volume 24, Number 4, August 2019

A Holistic Energy-Efficient Approach for a Processor-Memory System

Feihao Wu, Juan Chen*, Yong Dong, Wenxu Zheng, Xiaodong Pan,
Yuan Yuan, Zhixin Ou, and Yuyang Sun

Abstract: Component overclocking is an effective approach to speed up the components of a system to realize a higher program performance; it includes processor overclocking or memory overclocking. However, overclocking will unavoidably result in increase in power consumption. Our goal is to optimally improve the performance of scientific computing applications without increasing the total power consumption for a processor-memory system. We built a processor-memory energy efficiency model for multicore-based systems, which coordinates the performance and power of processor and memory. Our model exploits performance boost opportunities for a processor-memory system by adopting processor overclocking, processor Dynamic Voltage and Frequency Scaling (DVFS), memory active ratio adjustment, and memory overclocking, according to different scientific applications. This model also provides a total power control method by considering the same four factors mentioned above. We propose a processor and memory Coordination-based holistic Energy-Efficient (CEE) algorithm, which achieves performance improvement without increasing the total power consumption. The experimental results show that an average of 9.3% performance improvement was obtained for all 14 benchmarks. Meanwhile the total power consumption does not increase. The maximal performance improvement was up to 13.1% from dedup benchmark. Our experiments validate the effectiveness of our holistic energy-efficient model and technology.

Key words: processor overclocking; memory overclocking; performance boost; total power control; energy efficiency

1 Introduction

Processor overclocking can greatly speed up processors and consequently reduce the execution times of programs; however, it results in huge increase in power consumption. Developing an approach to keep the total power consumption from increasing when components are overclocked will favor both performance and energy, which will improve energy efficiency. An adaptive overclocking controller that dynamically applies the overclocking technique based on the application characteristics was proposed in Ref. [1]. From the results, the authors concluded that processor overclocking can better reduce energy consumption and energy delay product compared with Dynamic Voltage and Frequency Scaling (DVFS) and lower voltage techniques. According to their viewpoint, although the DVFS scheme consumes much less power than the baseline scheme, it severely degrades the performance due to the lowest clock frequency. Thus, it in turn increases the energy consumption of the microprocessor and the overall system. This means the energy efficiency can be improved from the angle of

• Feihao Wu, Juan Chen, Yong Dong, Wenxu Zheng, Xiaodong Pan, Yuan Yuan, Zhixin Ou, and Yuyang Sun are with the College of Computer, National University of Defense Technology, Changsha 410073, China. E-mail: wufeihao16@nudt.edu.cn; juanchen@nudt.edu.cn; yongdong@nudt.edu.cn; zhengwenxu17@nudt.edu.cn; panxiaodong17@nudt.edu.cn; yuanyuan@nudt.edu.cn; zhixinou@foxmail.com; sunyuyang980602@163.com.

∗ To whom correspondence should be addressed.

Manuscript received: 2018-05-17; accepted: 2018-06-15

component overclocking. Moreover, Ref. [1] focuses on the overall system energy, which motivates us to consider coordinating processor and memory, rather than merely processor, to improve the energy efficiency of the processor-memory system. This way, memory overclocking can be considered as another method for performance boost.

Our goal is to improve performance through component overclocking while controlling total power consumption from increasing. We consider combining these two aspects as our goal because of the following: First, nowadays, high power consumption is one of the major challenges in developing next-generation exascale supercomputer systems[2], which are associated with problems of system reliability or stability. Therefore, reducing or limiting power consumption is necessary. Second, component overclocking technique can improve performance, but will result in huge increase in power consumption, negatively affecting system reliability. Considering that power consumption has been a big problem and overclocking will increase the power consumption of the corresponding components, it is important to control the total power to reduce such risk. Third, energy saving is inevitable once our goal is achieved, as we will improve performance and reduce power consumption at the same time.

Several studies have been conducted to obtain a better performance-power tradeoff for the processor[3,4], but they do not consider the memory aspect. However, it is more beneficial to propose a holistic approach for energy-efficient computing than address either processor or memory in an isolated manner. This is because an isolated approach sometimes cannot achieve maximal energy efficiency. Considering a system with multiple adaptive components, maximizing the performance of a given component may not maximize the performance of the system. To maximize the system performance, the parameter space needs to be hand-tuned. Moreover, existing algorithms cannot be mechanically combined for multiple components. It is complex to keep total power consumption from increasing for performance-optimal algorithms across different components.

Many studies aim to develop a balanced energy-efficient system. In such studies, energy efficiency, which is measured by workload/Joule, with the unit of FLoating-pointing OPerations per Joule (FLOP/J), is improved by maximizing component utilization so that no component in the system is a bottleneck. This approach has been applied to data-intensive systems that balance processor utilization and storage I/O[5–7]. This approach is also appropriate for process-memory systems. In Refs. [8–12], researchers reasonably allocated power to processor and memory to improve performance within the power limits. Although these methods can realize good effects, none of them use overclocking techniques. It is important to note that overclocking can be used after other energy efficiency methods have been adopted. That is, we can further improve performance through processor or memory overclocking irrespective of how much other methods have improved performance; meanwhile, total power is not increased; therefore, energy can be reduced and energy efficiency can be improved. Thus, although our approach improves performance by only 9.3% on average, it is still better than the current approaches for improving energy efficiency.

Our method can improve the energy efficiency of the whole system, including processor and memory, by boosting performance and controlling the average power consumption. We achieve a sustaining performance boost by processor overclocking[1,13] and memory overclocking[14]. Meanwhile, we sustain an invariable total power by transferring power between memory side and processor side, where power transfer from memory to processor is realized by scaling down memory active ratio, and that from processor to memory is realized by scaling down the voltage/frequency of processor. We build a processor-memory energy efficiency model to describe the above performance-power tradeoff. Furthermore, we propose a processor and memory Coordination-based holistic Energy-Efficient (CEE) algorithm to implement the energy efficiency of the whole system.

The contributions of our article include the following:

(1) Categorization of component overclocking scenarios. We discover that two categories exist: processor overclocking and memory overclocking. Each of them can distinctly improve performance by overclocking, depending on the applications: CPU-intensive or memory-intensive applications. However, not all memory-intensive applications can benefit from memory overclocking, as performance improvement is mainly determined by how serious memory traffic is. Furthermore, we provide underlying processor-memory power transfer methods by a processor-memory energy efficiency model.

(2) Critical overclocking boundary for processor and memory. We find a clear linkage between processors' and memory modules' overclocking levels and categories of component overclocking scenarios. This linkage indicates the proper range in which memory overclocking works, by carefully checking the actual memory traffic level, for overclocking to desirably boost performance without any power consumption increase.

(3) Processor and memory coordination-based holistic energy-efficient algorithm. We show the overclocking boundary for a given application and its actual tested memory bandwidth. Based on this category, we determine the overclocking component (processor or memory) and a corresponding power-saving method (scaling down memory active ratio or processor DVFS). This algorithm can greatly improve performance and can accurately control the total power from increasing for all 14 benchmarks.

The rest of this paper is organized as follows: Section 2 introduces the basic idea; Section 3 illustrates the proposed processor-memory energy efficiency model and CEE algorithm; Section 4 describes our experimental methodologies and the experimental results; Section 5 shows related work; and Section 6 concludes the paper.

2 Basic Idea

Most scientific applications can be divided into CPU-intensive or memory-intensive. For CPU-intensive applications, we can overclock processors to achieve higher performance. To transfer the consequent power increase from the processor to memory, we can reduce memory power consumption by carefully scaling down memory active ratio with no performance degradation. For memory-intensive applications, we can overclock memory frequency to reduce execution time. To transfer the consequent power increase from memory to the processor, we can reduce the processor power consumption by carefully scaling down the processor frequency without any performance loss. As a result, we improve performance without increasing total power consumption for the two types of scientific applications.

There are three issues in the above method that we need to solve.

(1) First, memory overclocking cannot improve performance in some memory-intensive applications with very low memory traffic. Therefore, we cannot appropriately categorize component overclocking scenarios according to CPU-intensive or memory-intensive types. Here memory traffic represents the actual memory bandwidth measured during an application run. Our solution to this problem is that we extract those memory-intensive applications with lower memory traffic as an independent part, called weakly memory-intensive, also referred to as the "in-between". This means that memory-intensive application is divided into strongly memory-intensive and weakly memory-intensive for correct categorization of component overclocking scenarios. As Fig. 1 illustrates, scientific applications are divided into two types. The first type includes CPU-intensive and weakly memory-intensive applications, which is marked as a dotted box, and the other type refers to strongly memory-intensive applications.

(2) The second issue is lack of a clear metric to find the overclocking boundary for the above categorization.

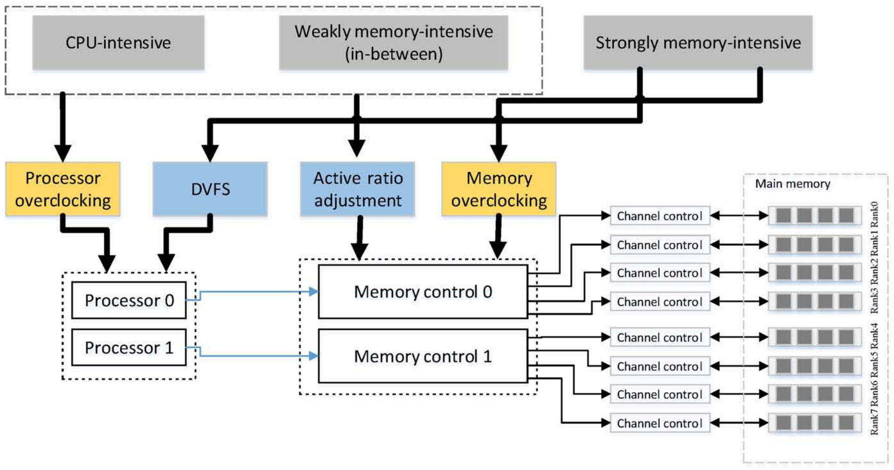

Fig. 1 Coordinated energy efficiency architecture.

We need to determine a proper range in which memory overclocking really works. Our solution to this problem is to find a metric to measure whether memory overclocking is beneficial or not. This metric is based on the actual memory traffic and memory bandwidth for a given program.

Memory traffic has a great influence on memory latency, which will affect performance. When the memory traffic is far below memory bandwidth, memory latency increases slowly as memory traffic increases. However, memory latency increases sharply when memory traffic is higher than a certain value.

For high memory traffic applications, i.e., strongly memory-intensive applications, memory overclocking greatly reduces memory latency, as shown in Fig. 2. We observe the memory latency change when memory traffic is 100 GB/s. Memory latency reduces from 140 ns to 120 ns when memory frequency increases from 1866 MHz to 2133 MHz. Figure 3 explains this case via a plot of memory bandwidth change. When memory frequency increases from 1866 MHz to 2133 MHz with memory active ratio fixed to one, memory bandwidth will increase from 111 GB/s to 124 GB/s. Memory bandwidth increase also results in memory latency reduction. Meanwhile, for CPU-intensive applications, processor overclocking also reduces memory latency. Figure 4 shows how memory latency is influenced by processor frequency in various memory active ratios.

(3) The third issue is how to keep total power from increasing based on the above two overclocking categories. Our solution is illustrated in Fig. 1. For CPU-intensive and weakly memory-intensive applications, we overclock processor for higher performance and reduce memory power consumption by scaling down memory active ratio. For strongly memory-intensive applications, we overclock memory frequency for performance boost and reduce processor power consumption by processor DVFS.

We define the *memory active ratio* as the number of memory ranks in the active state normalized to the total number of ranks in the memory system. For example, if we keep four out of eight ranks in the active state, the active ratio is 0.5. The memory active ratio is kept constant when a program is running. Figure 1 shows the memory scheduler, which controls the memory to perform a program under a given memory active ratio. Memory overclocking is also set by the memory scheduler. Multiple memory channels can sustain multiple memory requests at a given time. In our real platform, one memory rank corresponds to one memory channel.

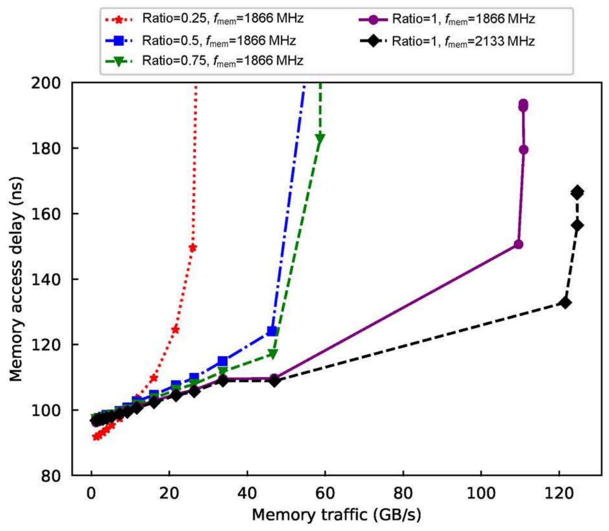

Fig. 2 Change of memory access delay with memory traffic.

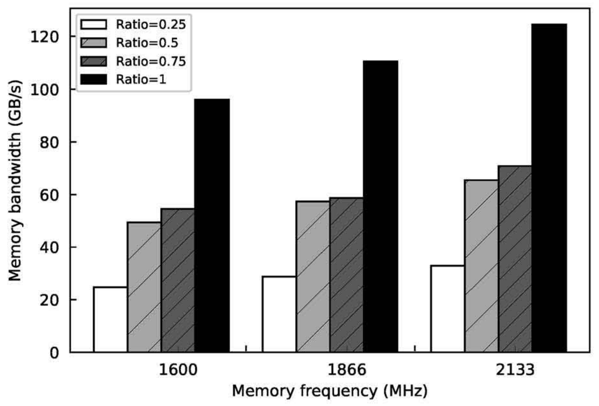

Fig. 3 Memory bandwidth increase with memory frequency under different memory active ratios.

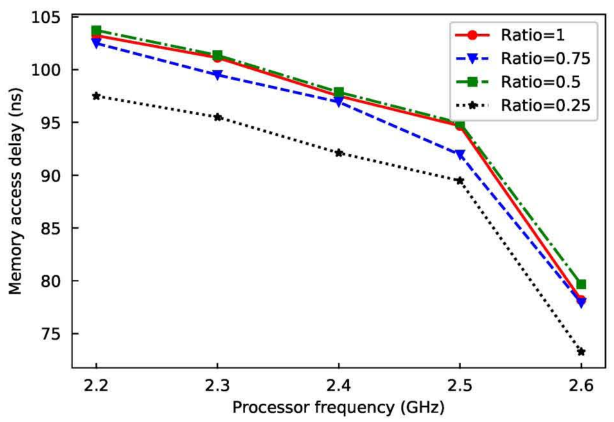

Fig. 4 Relationship between memory access delay and processor frequency. (Normal processor frequency is 2.4 GHz. Processor overclocking frequencies are 2.5 GHz and 2.6 GHz. Current memory frequency is 1866 MHz. Memory traffic is nearly 0 GB/s.)

Because CPU-intensive and weakly memory-intensive applications have low memory traffic, scaling down memory active ratio hardly influences memory latency. Figure 2 shows the relationship between memory latency and memory active ratio in various memory traffic situations.

When memory traffic is high, enough idle time is usually available for processor to reduce its frequency with little performance loss.

Next, we present some examples to validate the effectiveness of the above solutions.

For CPU-intensive applications, we adopt processor overclocking to boost performance and scale down memory active ratio to save memory power consumption. Eight of our benchmarks can be considered to be CPU-intensive. For example, fluidanimate uses 8.3% processor overclocking to reduce the time by 5.9%. Meanwhile, we scale down 50% memory active ratio to save a large amount of memory power consumption by 42.1%, because of the huge increase in power by processor overclocking, we save 1.5% of the total power consumption. The fewer the memory ranks in the active state, the less memory power consumption. Figure 5 shows the distribution of memory power consumption on different memory active ratios. We notice that scaling down memory active ratio has nearly no impact on these applications' performance in this situation.

For weakly memory-intensive applications (also called in-between), memory traffic is not high enough so that memory overclocking cannot significantly boost their performance. Instead, processor overclocking can improve the performance in this situation. For this in-between category, we have four benchmarks (canneal, facesim, ferret, and streamcluster). For example,

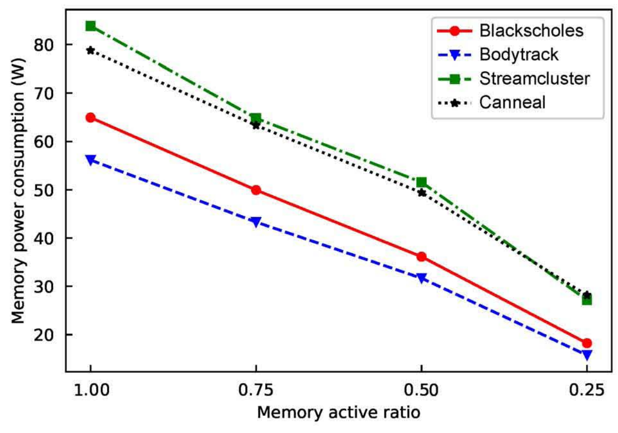

Fig. 5 Relationship between memory power and memory active ratio.

streamcluster performance cannot be improved by memory overclocking, and more power will yet be consumed. We use processor overclocking to boost performance and scale down memory active ratio to reduce memory power consumption. Streamcluster uses 8.3% processor overclocking to reduce the time by 5%. Meanwhile, we scale down 50% of memory active ratio to save 38.2% memory power consumption. Overall, 1.8% of the total power consumption is reduced.

For strongly memory-intensive applications, we use memory overclocking for performance improvement and scale down processor clock frequency for processor power reduction. We have six benchmarks which belong to memory-intensive category. For example, randomAccess uses 14.2% memory overclocking to reduce the time by 8.4%. Meanwhile, we reduce processor power consumption by 10.5%. Because of power increase by memory overclocking, the total power consumption is reduced by 1.4%.

3 Model and Algorithm

In this section, we present the processor-memory energy efficiency model and processor and memory coordination-based holistic energy-efficient algorithm.

3.1 Processor-memory energy efficiency model

Table 1 lists all the parameters and their meanings in our model. Execution time T and total power P both depend on $f_{\text{CPU\_over}}$, f_{DVFS}, r_{mem}, and $f_{\text{mem\_over}}$. For a given application, *memory traffic ratio* represents the ratio of memory traffic to memory bandwidth. We define α as the threshold for the memory traffic ratio.

The objective of our model is shown in Eq. (1).

$$\begin{aligned} \min \quad & T = T(f_{\text{CPU\_over}}, f_{\text{DVFS}}, r_{\text{mem}}, f_{\text{mem\_over}}), \\ \text{s.t.} \quad & \Delta P^{+} + \Delta P^{-} \leqslant 0, \\ & \Delta P^{+} \geqslant 0, \\ & \Delta P^{-} \leqslant 0 \end{aligned} \tag{1}$$

Here, ΔP^{+} represents the power increase in the total power consumption compared to the baseline, where the baseline situation represents the condition where both processor and memory frequencies are at normal level and memory active ratio is 1. Similarly, ΔP^{-} represents the power decrease in the total power consumption compared to the baseline.

When memory bandwidth determined by the real hardware is reduced by scaling down memory active ratio, memory traffic ratio will increase. According to the actual memory traffic ratio, we divide the

Table 1 Parameters and corresponding meanings in our model.

Parameter	Meaning
$f_{\text{CPU\_over}}$	Processor overclocking frequency
$[F_{\text{CPU\_over\_1}}, \dots, F_{\text{CPU\_over\_}K}]$	Processor overclocking frequency range
f_{DVFS}	Processor DVFS frequency
$[F_{\text{DVFS\_1}}, \dots, F_{\text{DVFS\_}M}]$	Processor DVFS frequency range
$f_{\text{mem\_over}}$	Memory overclocking frequency
$[F_{\text{mem\_over\_1}}, \dots, F_{\text{mem\_over\_}N}]$	Memory overclocking frequency range
r_{mem}	Memory active ratio
b	Memory traffic
$\text{BW} = \langle \text{bw}_1, \text{bw}_2, \dots, \text{bw}_m \rangle$	m-level memory bandwidth
α	The threshold for the ratio of memory traffic to memory bandwidth
T	The application execution time
P	The total power consumption for the processor and main memory
$\Delta P^+, \Delta P^-$	The total power difference between current power and the baseline
P_0, P_0^{CPU}, P_0^{mem}	The baseline for the total power consumption, processor power, and memory power, where processor and memory frequencies are both normal level, and memory active ratio is 1.

applications into two categories: those whose actual memory traffic ratio is less than α and those with actual memory traffic ratio not less than α.

When actual memory traffic ratio is less than α, such as in CPU-intensive and weakly memory-intensive applications, the reduction of T comes from the increase of $f_{\text{CPU\_over}}$, as shown in Eq. (2a). The constraint condition in Eq. (1) guarantees that total power consumption P does not increase. $\Delta P^+ + \Delta P^- \leqslant 0$ can be substituted by Eq. (2b).

When actual memory traffic ratio is not less than α, such as in strongly memory-intensive applications, the reduction of T arises from the increase of $f_{\text{mem\_over}}$, as shown in Eq. (3a). The constraint condition in Eq. (1) guarantees that the total power consumption P does not increase. $\Delta P^+ + \Delta P^- \leqslant 0$ can be substituted by Eq. (3b).

$$T \downarrow = T(f_{\text{CPU\_over}} \uparrow, f_{\text{DVFS}}, r_{\text{mem}} \downarrow, f_{\text{mem\_over}}) \tag{2a}$$

$$\Delta P^+ + \Delta P^- = P(f_{\text{CPU\_over}} \uparrow, f_{\text{DVFS}}, r_{\text{mem}} \downarrow, f_{\text{mem\_over}}) - P_0 \tag{2b}$$

$$T \downarrow = T(f_{\text{CPU\_over}}, f_{\text{DVFS}} \downarrow, r_{\text{mem}}, f_{\text{mem\_over}} \uparrow) \tag{3a}$$

$$\Delta P^+ + \Delta P^- = P(f_{\text{CPU\_over}}, f_{\text{DVFS}} \downarrow, r_{\text{mem}}, f_{\text{mem\_over}} \uparrow) - P_0 \tag{3b}$$

3.2 Algorithm

We propose a processor and memory coordination-based holistic energy-efficient algorithm in Algorithm 1. For a given application, this algorithm obtains nearly optimal $f_{\text{CPU\_over}}$, f_{DVFS}, r_{mem}, $f_{\text{mem\_over}}$ for performance boost without power consumption increase.

Input parameters consist of m-level memory bandwidth $\text{BW} = \langle \text{bw}_1, \text{bw}_2, \dots, \text{bw}_m \rangle$, processor and memory frequency scaling ranges, and memory

Algorithm 1 Processor and memory coordination-based holistic energy-efficient algorithm

```
Require:
    $m$-level memory bandwidth
    $\text{BW} = \langle \text{bw}_1, \text{bw}_2, \dots, \text{bw}_m \rangle$; memory traffic ratio
    threshold $\alpha$; $[F_{\text{CPU\_over\_1}}, \dots, F_{\text{CPU\_over\_}K}]$;
    $[F_{\text{DVFS\_1}}, \dots, F_{\text{DVFS\_}M}]$, $[F_{\text{mem\_over\_1}}, \dots, F_{\text{mem\_over\_}N}]$;
Ensure:
    Processor overclocking frequency $f_{\text{CPU\_over}}$, or processor
    DVFS frequency $f_{\text{DVFS}}$, memory overclocking frequency
    $f_{\text{mem\_over}}$, memory active ratio $r_{\text{mem}}$,
1: Obtain parameters memory traffic $b$, $P_0^{\text{CPU}}$, $P_0^{\text{mem}}$;
2: if $b/\text{bw}_m < \alpha$ then
3:     /* CPU-intensive and weakly memory-intensive */
       $\frac{b}{\text{bw}_k} \leftarrow \max\limits_{1 \leqslant i \leqslant m} \{\frac{b}{\text{bw}_i} | \frac{b}{\text{bw}_i} < \alpha\}$;
4:     $r_{\text{mem}} = k/m$;
5:     Scale down memory active ratio to $r_{\text{mem}}$;
6:     Update memory power
       $P^{\text{mem}}(f_{\text{CPU\_over}}, f_{\text{DVFS}}, r_{\text{mem}}, f_{\text{mem\_over}})$;
7:     Calculate power savings
       $\Delta P^- = P^{\text{mem}}(f_{\text{CPU\_over}}, f_{\text{DVFS}}, r_{\text{mem}}, f_{\text{mem\_over}}) - P_0^{\text{mem}}$;
8:     Find the maximum $F_{\text{CPU\_over\_}k} (1 \leqslant k \leqslant K)$, which
       satisfies $\Delta P^+ + \Delta P^- \leqslant 0$;
9:     Output $f_{\text{CPU\_over}}$ and $r_{\text{mem}}$;
10: end if
11: if $b/\text{bw}_m \geqslant \alpha$ then
12:     /* Strongly memory-intensive */ Determine $f_{\text{mem\_over}}$,
        $f_{\text{mem\_over}} \in \{F_{\text{mem\_over\_1}}, \dots, F_{\text{mem\_over\_}N}\}$;
13:     Set memory overclocking frequency to $f_{\text{mem\_over}}$;
14:     Update memory power
        $P^{\text{mem}}(f_{\text{CPU\_over}}, f_{\text{DVFS}}, r_{\text{mem}}, f_{\text{mem\_over}})$;
15:     Calculate power increase
        $\Delta P^+ = P^{\text{mem}}(f_{\text{CPU\_over}}, f_{\text{DVFS}}, r_{\text{mem}}, f_{\text{mem\_over}}) - P_0^{\text{mem}}$;
16:     Find the maximum $F_{\text{DVFS\_}k} (1 \leqslant k \leqslant M)$, which satisfies
        $\Delta P^+ + \Delta P^- \leqslant 0$;
17:     Output $f_{\text{DVFS}}$ and $f_{\text{mem\_over}}$;
18: end if
```

traffic ratio threshold. Here, bw_m represents the default memory bandwidth in the baseline situation; bw_1 represents the minimal memory bandwidth when scaling down as much memory active ratio as possible; and m is the number of memory channels. In our experiment, the number of memory channels equals the number of memory ranks. The element $\mathrm{bw}_k (k = 1, 2, \ldots, m)$ represents the memory bandwidth when memory active ratio is k/m.

The outputs for this algorithm include processor and memory frequency scaling values and memory active ratio. Processor frequency is either overclocked or scaled down. Memory frequency is either overclocked or kept unchanged. According to the memory traffic ratio threshold provided by our algorithm, our algorithm achieves near-optimal memory active ratio, which determines an appropriate level from m-level memory bandwidth.

The algorithm is divided into two parts. First, we need to obtain three parameters, memory traffic b, P_0^{CPU}, P_0^{mem} by running an application and conducting some performance profile and power measurements. The initial processor power and memory power are separately measured. Characterizing the applications via profiling will not cause a limitation for our approach, because in a supercomputing center, most scientific computing applications often run multiple times. Even if a profile-based approach consumes a large amount of time on profiling data, we can still benefit from the later process, which is running this program repeatedly.

Second, according to the relationship between b/bw_m and α, the algorithm is divided into two branches: Steps 2–9 and Steps 11–17. In the former branch, for CPU-intensive applications, Step 3 obtains the optimal memory bandwidth bw_k, and Step 4 obtains the corresponding memory active ratio r_{mem}. Power saving comes from memory side because memory active ratio is scaled down (Step 7). To satisfy the power constraint condition $\Delta P^+ + \Delta P^- \leqslant 0$, the maximal processor overclocking frequency is calculated in Step 8. Finally, our algorithm outputs the data $f_{\mathrm{CPU\_over}}$ and r_{mem}.

The other branch for the second part is for memory-intensive applications. In Step 12, we find a memory overclocking frequency from $[F_{\mathrm{mem\_over\_1}}, \ldots, F_{\mathrm{mem\_over\_}N}]$. Step 14 updates the memory power after adopting memory overclocking. Power increase comes from the memory side (Step 15). To satisfy the power constraint condition $\Delta P^+ + \Delta P^- \leqslant 0$, the maximal processor DVFS frequency is calculated in Step 16. Finally, our algorithm outputs the data f_{DVFS} and $f_{\mathrm{mem\_over}}$.

In the following section, we validate the effectiveness of our algorithm via experiments.

4 Experiments

In this section, we introduce our experimental platform and results.

4.1 Experimental platform

We used some parallel benchmarks from two benchmark suites, which included all the benchmarks in PARSEC[15] and part of benchmarks in HPCC[16]. All these benchmarks are listed in Table 2. They are the representatives for many important scientific applications and are divided into two types: CPU-intensive and memory-intensive. Among memory-intensive types, there are four benchmarks that belong to weakly memory-intensive, and the other two belong to strongly memory-intensive.

All benchmarks were executed with up to 20 threads. In our experimental environment, the maximum number of processor cores was 20. However, two benchmarks, fluidanimate and facesim, cannot be executed with 20 threads because it requires the number of threads to be the power of two. Therefore, the scale for these two benchmarks was 16. For all the multi-threaded benchmarks, each thread was mapped to one processor core.

We used a dual-socket Intel/Linux system with a

Table 2 Benchmarks used in this study.

Benchmark suite	Benchmark	Scale	Type
PARSEC	Blackscholes	20	CPU-intensive
PARSEC	Bodytrack	20	CPU-intensive
PARSEC	Dedup	20	CPU-intensive
PARSEC	Fluidanimate	16	CPU-intensive
PARSEC	Freqmine	20	CPU-intensive
PARSEC	Swaptions	20	CPU-intensive
PARSEC	x264	20	CPU-intensive
PARSEC	Vips	20	CPU-intensive
PARSEC	Canneal	20	Weakly memory-intensive
PARSEC	Facesim	16	Weakly memory-intensive
PARSEC	Ferren	20	Weakly memory-intensive
PARSEC	Streamcluster	20	Weakly memory-intensive
HPCC	Stream	20	Strongly memory-intensive
HPCC	RandomAccess	20	Strongly memory-intensive

PR3016GS2 motherboard, two Intel Xeon E5-2660V3 processors, and 128 GB, 2133 MHz, and DDR4 memory. This motherboard supports RAPL[17] for power measurement. The cores support DVFS among 15 status, from 1.2 GHz to 2.6 GHz, with a step of 0.1 GHz. The thermal design power for these processors is 105 W. To accurately measure the power results, we disabled the processor hyperthreading.

Overclocking technology is widely used in most current processors. Processor overclocking can effectively provide much greater speedup for processors, which boosts application performance. Similarly, memory overclocking can boost performance for memory access. In our CEE algorithm, processor overclocking and memory overclocking are adopted to improve performance. More importantly, the system reliability will not be greatly influenced through overclocking. The reasons are as follows: (1) There are several works in which overclocking technology was used[1, 13, 14]. In Ref. [1], system reliability was evaluated by measuring the normalized failures in time. The experimental results show that failure ratio is not greatly influenced by processor overclocking. (2) Currently, many researches[18, 19] adopt processor overclocking, memory overclocking, or both to improve system performance. Their actual test observations show that the system is reliable during overclocking. (3) Compared with previous research works, our CEE algorithm adopts processor overclocking with only two levels and memory overclocking with only one level. Therefore, processor overclocking and memory overclocking are strictly limited within a certain range, which furthermore guarantees system reliability when we use the CEE algorithm.

Nowadays, there are many studies on overclocking, and the main methods used in these studies to overclock processors are Turbo Boost technology[20], overclocking by increasing multiple frequencies[1], and simulated overclocking[13]. (1) Lo and Kozyrakis[20] used Turbo Boost technology for processors overclocking. Turbo technology improves performance through automatic processor overclocking. Its overclocking value is transparent to users, and users cannot set the desired overclocking value. Our method needs to set a specific overclocking frequency calculated by the CEE algorithm; thus, Turbo Boost technology cannot be used to implement our experiment. (2) Jang et al.[1] overclocked a processor by setting BIOS and increasing multiple frequencies. They used an Intel desktop processor and gigabyte motherboard. Intel desktop processors support processor overclocking by increasing multiple frequencies, which provides hardware support for dramatic/static processor frequency adjustments. Although this method can be used to set the desired overclocking value, to date, there is no suitable API for software-controlled dynamical overclocking of processors. Therefore, the processor can only be overclocked by a static method, like the BIOS, but cannot be dynamically adjusted. However, Intel's server series chips, which are used in our experiment, do not support multiple frequencies adjustment, as shown in Table 3. Therefore, we cannot use the BIOS to achieve static processor overclocking as in Ref. [1]. (3) Rubio et al.[13] used DVFS technology to simulate overclocking, using a Pentium M processor. The normal frequency range of this processor is 600 MHz to 2 GHz, while in the simulated overclocking, 1.8 GHz and 2.0 GHz were used as the normal frequency and overclocking frequency, respectively. Our experiments also included simulated overclocking. For the processor we use, the processor overclocking range was 2.6–3.1 GHz. In our simulated overclocking, we assumed that 2.4 GHz was the nominal frequency while 2.5 GHz and 2.6 GHz were overclocking frequencies. At present, simulated overclocking is a reliable method to use in our platform. Table 4 lists the processor overclocking frequency range used in our experiment.

We used a 128 GB DDR4 memory in our experiment. The normal memory frequency was 2133 MHz. The actual memory overclocking frequency range was [2133 MHz, ..., 3000 MHz]. As for Intel Xeon E5-2600V3, we used 1866 MHz as the normal memory

Table 3 Different processes and corresponding overclocking technologies.

CPU maker	Model	Turbo support	Overclocking by BIOS
Intel	Server (E5 2660V3)	Yes	No
	Desktop (i7-6700HQ)	Yes	Yes
AMD	Server (Opteron X3000)	Yes	N\A
	Desktop (Ryzen 7 1700)	Yes	Yes

Table 4 The parameter values in our experiment.

Parameter name	Parameter value
$[F_{\text{CPU\_over\_1}}, \ldots, F_{\text{CPU\_over\_}K}]$	$[2.5\,\text{GHz}, 2.6\,\text{GHz}]$, $K = 2$
$[F_{\text{DVFS\_1}}, \ldots, F_{\text{DVFS\_}M}]$	$[1.2\,\text{GHz}, \ldots, 2.4\,\text{GHz}]$, $M = 13$
$[F_{\text{mem\_over\_1}}, \ldots, F_{\text{mem\_over\_}N}]$	$[2133\,\text{MHz}]$, $N = 1$
α	0.5

frequency, rather than 2133 MHz, because Intel Xeon E5-2600V3 does not support memory frequency higher than 2133 MHz. The memory overclocking frequency range was 1866–2133 MHz, which is also given in Table 4.

To get some insight on the distribution of different levels of memory traffic for each benchmark, we used vtune[21] to count the information about memory traffic every 1 ms. Then, we made a histogram of the memory traffic, which divided the ranges of memory traffic into bins. For example, for blackscholer benchmark, there were five memory traffic bins, 2 GB/s, 5 GB/s, 8 GB/s, 11 GB/s, and 14 GB/s, as shown in Fig. 6. The number of bins and the size of each bin depended on benchmarks. Vtune counts the total elapsed time in each bin. Then, we plotted a bar graph to show the relative sizes of the bins. For example, Fig. 6 shows the histogram of memory traffic for blackscholer benchmark. For all the memory traffic bins, we ignored bins that had 0.5 s or less elapsed time. Then, we chose the maximal one from the rest of memory traffic bins as the memory traffic of this benchmark. For blackscholer benchmark, 8 GB/s, 11 GB/s, and 14 GB/s bins were ignored. Finally, we took 5 GB/s as the memory traffic for blackscholer benchmark. We used powergov[22] to measure processor power consumption and memory power consumption. The powergov uses the Intel RAPL technology[17] to profile processor and memory power consumptions. We used Intel® Memory Latency Checker (MLC)[23] to measure memory bandwidth and memory access delay under the various memory traffic conditions.

We needed to achieve processor frequency under a given power. Lefurgy et al.[24] and Raghavendra et al.[25] pointed out that the processor power is approximately linear to the processor clock frequency: $p_p = a \times f_p + b$. For different benchmarks, there were some changes in parameters a and b. To obtain more accurate parameters a and b, one benchmark was executed twice. Then, two groups of processor frequency and processor power values could determine the values of parameters a and b. According to this processor power model and the determined parameters, we could obtain processor frequency under a given power consumption.

After processor frequency was adjusted, memory power was nearly unchanged. In our experiment, we measured the memory power on a real hardware under several processor frequencies and found memory power hardly changed by processor frequency scaling, as shown in Fig. 7. We assumed memory power was constant.

4.2 Experiment results

In this section, we introduce our experimental results and analyses. All the experimental data were obtained via measurements conducted at least three times. We took the average of these three values as the final result for each experimental data. Some slight fluctuations existed because of system factors, such as cache. To eliminate such interference, we needed to more measurement runs.

4.2.1 Overall results

Our CEE algorithm can greatly improve performance. As Fig. 8 shows, the average performance improvement was up to 9.3% for all the 14 benchmarks, and the maximal performance improvement was up to 13.1% for dedup benchmark.

Our CEE algorithm can guarantee no increase in total power consumption, as shown in Fig. 9. For

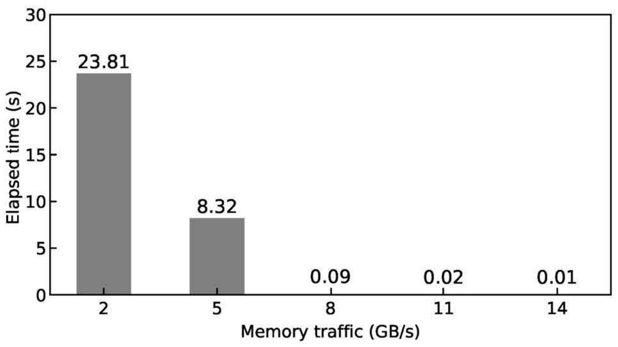

Fig. 6 The blackscholer memory traffic.

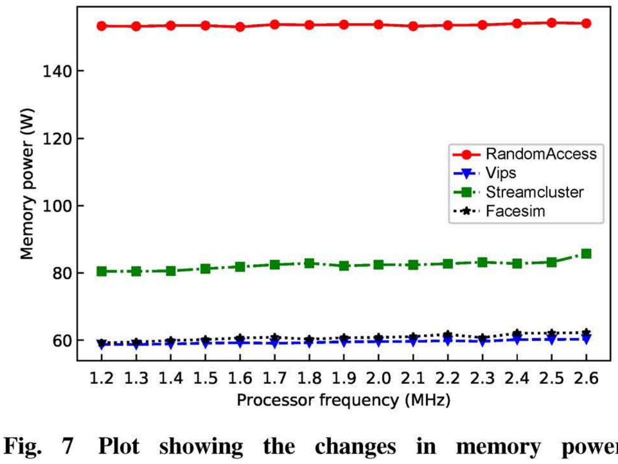

Fig. 7 Plot showing the changes in memory power consumption for all the benchmarks; the changes were insignificant, irrespective of if processor DVFS or processor overclocking was used.

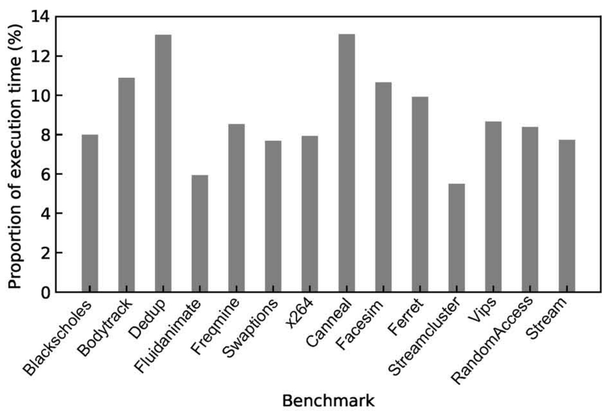

Fig. 8 Performance improvement by CEE algorithm.

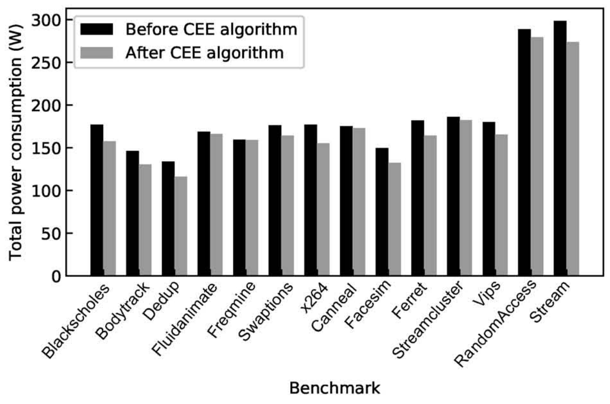

Fig. 9 Total power consumption before and after applying CEE algorithm.

some benchmarks, it greatly reduced the total power consumption. This is because scaling down memory active ratio by closing memory ranks can save a large amount of memory power consumption. We noticed that limited processor overclocking frequency stopped the power increase from reaching the upper bound of our model. The higher the processor overclocking frequency, the more the performance boosts.

As Fig. 1 shows, we considered four kinds of energy efficiency methods, as well as our CEE algorithm, in our model. For three types of applications, different energy efficiency methods were adopted to improve their performance and control power consumption. Table 5 lists the corresponding energy efficiency methods for each benchmark.

Our approach did not increase the total power consumed, as shown in Fig. 9. For some applications, it reduced power prominently. This is because closing memory rank can save a lot of power and we cannot increase processor frequency unlimitedly. If we can raise the processor overclocking frequency, we can get more performance boost.

Table 5 Energy efficiency methods adopted for each benchmark according to CEE algorithm.

Benchmark	Energy efficiency method for improving performance	Energy efficiency method for power control
Blackscholes	Processor overclocking	Active ratio adjustment
Bodytrack	Processor overclocking	Active ratio adjustment
Dedup	Processor overclocking	Active ratio adjustment
Fluidanimate	Processor overclocking	Active ratio adjustment
Freqmine	Processor overclocking	Active ratio adjustment
Swaptions	Processor overclocking	Active ratio adjustment
x264	Processor overclocking	Active ratio adjustment
Vips	Processor overclocking	Active ratio adjustment
Canneal	Processor overclocking	Active ratio adjustment
Facesim	Processor overclocking	Active ratio adjustment
Ferret	Processor overclocking	Active ratio adjustment
Streamcluster	Processor overclocking	Active ratio adjustment
Stream	Memory overclocking	DVFS
RandomAccess	Memory overclocking	DVFS

For different applications, we used different technologies. The technologies adopted for each benchmark are listed in Table 5.

4.2.2 Performance results

This subsection shows the performance results of our CEE algorithm. The performance results are divided into three parts according to application type.

(1) CPU-intensive applications

For CPU-intensive benchmarks, such dedup, canneal, processor overclocking frequency can reduce CPU time and memory latency.

Figure 10 shows the memory bandwidth of the real hardware and memory traffic for each benchmark.

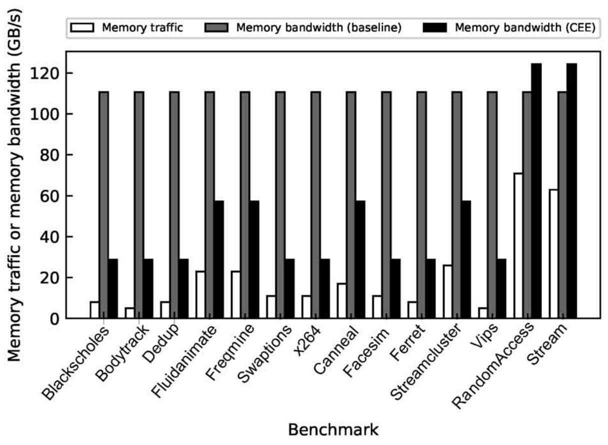

Fig. 10 Memory bandwidth reduction via CEE algorithm; memory bandwidth was greatly reduced, and the reduced memory bandwidth was still much higher than memory traffic.

The memory bandwidth (baseline), indicated by the darker column bar, is fixed and does not depend on applications. The column bar marked as memory traffic is much lower than memory bandwidth.

According to the memory active ratio threshold α, the CEE algorithm obtained the greater reduced memory bandwidth. As Fig. 10 shows, the rightmost column bar is much lower than the middle column bar for each group of bars. Moreover, reducing memory active ratio will cause little performance loss, as shown in Fig. 2. Furthermore, we tested only scaling down memory active ratio according to the CEE algorithm, and found that only scaling down memory active ratio will hardly influence the execution time, as shown in Fig. 11. We took two benchmarks, dedup and canneal, as an example. According to our CEE algorithm, the memory active ratios for these two benchmarks were scaled down to 0.25. When we only considered scaling down memory active ratio, without scaling down processor overclocking frequency, the execution times for this situation were not higher than those of the baseline and were even less, as shown in Fig. 11. Meanwhile, for the dedup and canneal benchmarks, power was saved by 31.3% and 23.9%, respectively, as shown in Fig. 12.

(2) Strongly memory-intensive applications

For strongly memory-intensive benchmarks, such as RandomAccess and Stream, the processor overclocking frequencies could not improve performance. According to the memory active ratio threshold $\alpha = 0.5$, the memory traffic values of RandomAccess and Stream were higher than the threshold by 14.2% and 7.0%, respectively. According to Fig. 13, the processor overclocking frequencies (2.5 GHz and 2.6 GHz) could

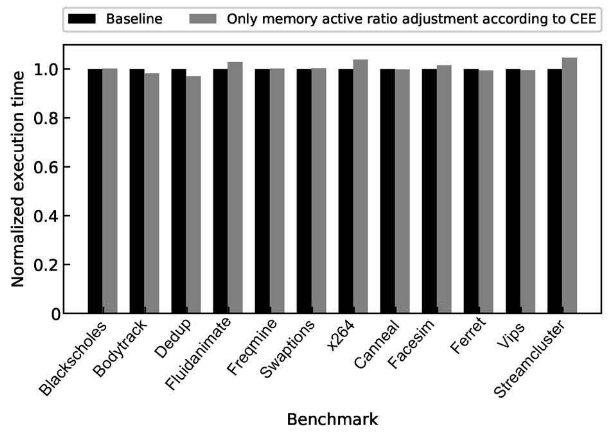

Fig. 11 Effect of scaling down memory active ratio without processor overclocking frequency on the execution time of CPU-intensive applications; execution time was hardly affected.

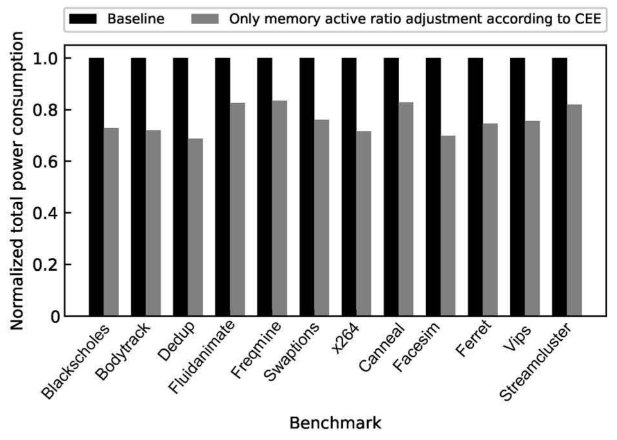

Fig. 12 Effect of scaling down memory active ratio without processor overclocking frequency on the total power consumption for CPU-intensive applications; total power consumption was greatly reduced.

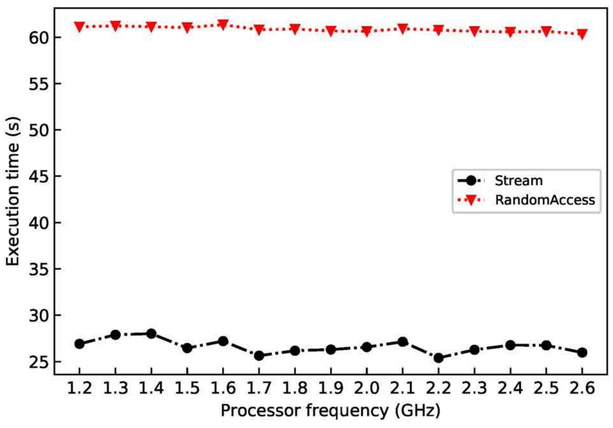

Fig. 13 Effect of processor overclocking on performance. For strongly memory-intensive applications, processor overclocking (2.5 GHz and 2.6 GHz) could not improve performance. In the entire processor DVFS range, there was nearly no change in the execution time, which means a lot of processor idle time was available in this case.

not improve performance. Meanwhile, from Fig. 13, we can conclude that there was a lot of CPU idle time. When the processor clock frequency was scaled down from 2.4 GHz to a lower level, even to 1.2 GHz, there was almost no loss in performance.

In this situation, memory overclocking frequency increased memory bandwidth and reduced memory access delay.

(3) Weakly memory-intensive (in-between) applications

Among all the memory-intensive applications, some applications will not benefit from memory overclocking and processor DVFS. In Fig. 14, the four leftmost benchmarks are weakly memory-intensive. As the figure shows, when we adopted memory overclocking

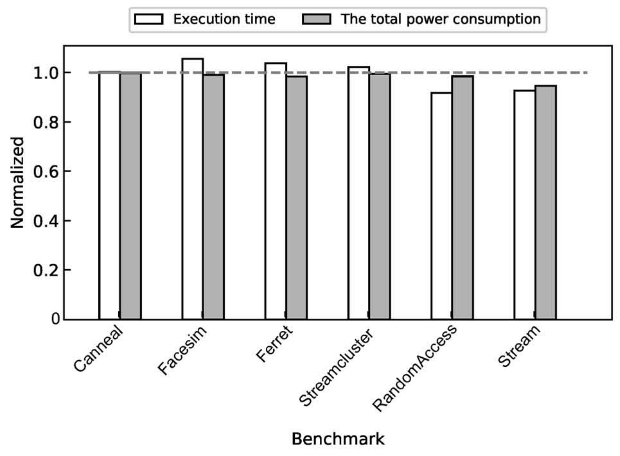

Fig. 14 Effect of memory overclocking and processor DVFS on performance. For weakly and strongly memory-intensive applications, memory overclocking and processor DVFS did not guarantee improving performance without increasing total power consumption. (The four leftmost benchmarks belong to weakly memory-intensive, and the two rightmost belong to strongly memory-intensive.)

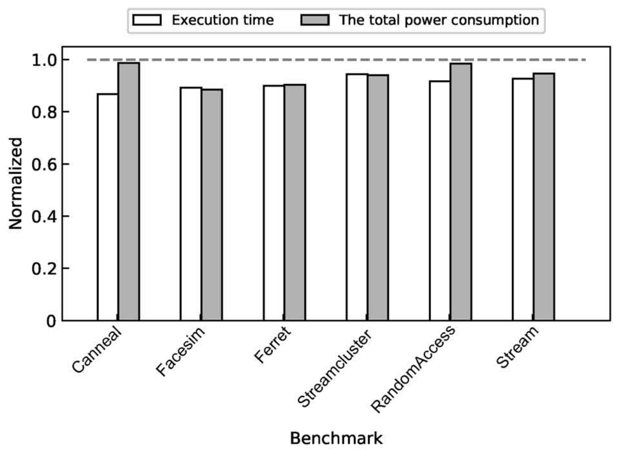

Fig. 15 Effects of processor and memory overclocking techniques on performance. For weakly memory-intensive applications, memory overclocking and processor DVFS were replaced by processor overclocking and memory active ratio scaling down techniques; performance was improved without increasing total power consumption. For strongly memory-intensive applications, memory overclocking and processor DVFS were adopted. (The four leftmost benchmarks belong to weakly memory-intensive, and the two rightmost belong to strongly memory-intensive.)

and processor DVFS, the average performance loss for these four benchmarks, i.e., canneal, facesim, ferret, and streamcluster, was up to 2.9%, and the maximum performance loss was up to 5.6%, which was for facesim benchmark. Therefore, memory overclocking and processor DVFS were not certain to improve performance and control the total power consumption from increasing for all memory-intensive applications (including weakly and strongly).

In our CEE algorithm, for weakly memory-intensive applications, we adopted processor overclocking and memory active ratio scaling down instead of memory overclocking and processor DVFS. As Fig. 15 illustrates, when we adopted processor overclocking and memory active ratio scaling down, the average performance improvement for these four benchmarks (canneal, facesim, ferret, and streamcluster) was 9.9%, and the maximal performance improvement was up to 13.2%, which was for canneal benchmark.

4.2.3 Total power results

To distinguish the power increase part and power decrease part in the total power consumption, we present a plot in Fig. 16. It can be clearly seen that the power consumption decrease column is sometimes much larger than the power consumption increase column. Overall, the CEE algorithm can ensure that $\Delta P^- + \Delta P^+ \leqslant 0$.

For CPU-intensive and weakly memory-intensive applications, first, we calculated ΔP^- according to the reduced memory active ratio. Then, we obtained ΔP^+ according to $\Delta P^- + \Delta P^+ \leqslant 0$. This means we can use processor overclocking to improve performance without total power increase.

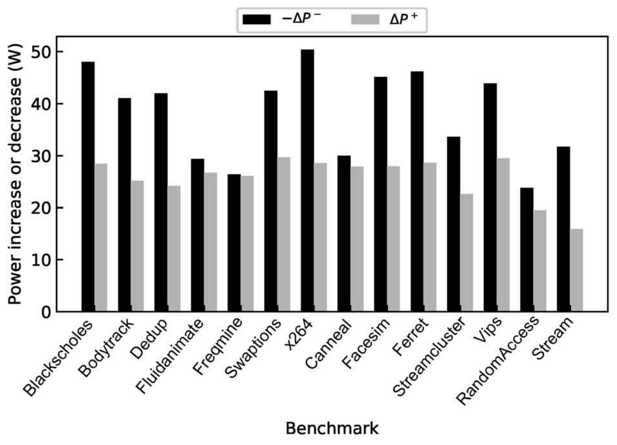

Fig. 16 Power increase part and power decrease part in the total power consumption.

For strongly memory-intensive applications, first, we obtained ΔP^+ according to the memory overclocking. Because there was only one level for memory overclocking frequency in our experiment as Table 4 shows, the power consumption increase ΔP^+ from memory overclocking frequency was not very high. Thus, we can use processor DVFS to save power and ensure $\Delta P^- + \Delta P^+ \leqslant 0$.

5 Related Work

High power and high energy consumption has become one of most important concerns in computer systems studies, particularly in high-performance computing[26, 27]. A large number of approaches are focused on this issue. Most low-power techniques save power consumption by scaling down processor clock frequency[28, 29]. Although these low-power techniques can reduce energy consumption, they will usually result in the performance loss. Another type of low-power approach[30] is not only scaling down processor clock frequency but also changing the number of active cores for power savings. In Refs. [31, 32], near-threshold computing technology was used to explore power-efficiency under performance constraints by a 128-core chip operating at the near-threshold region. In addition to reducing processor power consumption, Refs. [33–36] focused on how to reduce memory energy consumption.

Besides low-power techniques, researchers have also focused on power-constrained problems for computing nodes. In Refs. [8–12], power was reasonably allocated to CPU and memory for performance improvement within the power limits. The main idea was that for different applications, the power demands of processor and memory are different. According to applications' characteristics, the researchers allocated power to CPU and memory to satisfy their demand for performance and power. Furthermore, Refs. [37–40] focused on a cluster. When the power of a cluster is limited, first, the researchers needed to set the number of active nodes according to an application's scalability. Second, they needed to allocate the power to compute nodes, and also allocated the power to processor and memory in one node ultimately. Finally, they improved application's performance within the power constraints.

The work most related to ours is on developing balanced energy-efficient systems, which equally maximize the utilization of all components such that no component in the system is a bottleneck. Since components, e.g., processors and memory, are not power-proportional, they can achieve their highest energy efficiency when operated at maximum utilization. As a result, balanced systems also maximize performance for a given amount of energy because they eliminate bottlenecks and allocate the energy most efficiently. There are several works having a similar idea with the above[5–7]. However, they are focused on data-intensive systems that balance processor utilization and storage I/O. In these works, the platform to be balanced is designed; that is, few processors are paired with lots of disks, or a flash is used.

Our work applies a similar concept to computationally or memory-intensive High Performance Computing (HPC) applications. Basically, our algorithm recognizes when the power allocated to the processor is not being fully utilized (and thus is not at peak energy efficiency and is a bottleneck) and shifts that power to the memory system, which increases the utilization of the processor and the memory system, and also increases the performance for a given amount of energy. We use various existing dynamic power management mechanisms for CPUs and memory to shift the power. In addition, we use processor and memory overclocking techniques to realize performance improvement, which are seldom used. Lo and Kozyrakis[20] found that overclocking technology can result in higher energy efficiency in some situations.

Our research work focuses on how to coordinate the processor and memory in a multicore-based system for performance improvement while keeping total power consumption from increasing. Our research work is different from the previous works in the literature[8–12]. Ge et al.[10] focused on the problem of coordinated power allocation between processors and memory modules on power-bounded systems. They found that cross-component power coordination had great influence on application performance. We are also concerned about the coordination of processor and memory when overclocking and power-saving techniques are used. The difference between our work and the previous works is that those works are focused on power-bounded computing, while we study how to improve performance while ensuring no increase in total power consumption. We use overclocking for performance boost, but the previous researchers did not use overclocking technique. Lo and Kozyrakis[20] found that overclocking technology can result in higher energy efficiency in some situations.

6 Conclusion

Improving the energy efficiency of a processor-memory system is significant. This is because higher energy and power consumption has become one of biggest challenges for computer systems, particularly high-performance computing systems. For better energy efficiency, our objective is to boost performance while guaranteeing no increase in the total power consumption. This is quite different from reducing power consumption as much as possible with a little performance loss. We built a processor-memory energy efficiency model for multicore-based systems to coordinate processor and memory power distribution. Using four kinds of energy efficiency techniques—processor overclocking, processor DVFS, memory active ratio adjustment, and memory overclocking—our model explores performance boost opportunities and satisfies power control demand. We also propose a processor and memory coordination-based holistic energy-efficient algorithm to implement performance boost while the total power consumption does not increase. We provide detailed experimental results that validate the effectiveness of our model and algorithm.

Acknowledgment

The authors would like to thank to the funding from the National Key Research and Development Program of China (No. 2018YFB1003203), the Advanced Research Project of China (No. 31511010203), Open Fund from State Key Laboratory of High Performance Computing (No. 201503-02), and Research Program of NUDT (No. ZK18-03-10).

References

[1] H. B. Jang, J. Lee, J. Kong, T. Suh, and S. W. Chung, Leveraging process variation for performance and energy: In the perspective of overclocking, *IEEE Transactions on Computers*, vol. 63, no. 5, pp. 1316–1322, 2014.

[2] A. Subcommittee, Top ten exascale research challenges, Report, US Department Of Energy, USA, 2014.

[3] W. Wang, A. Porterfield, J. Cavazos, and S. Bhalachandra, Using per-loop CPU clock modulation for energy efficiency in openmp applications, in *Proc. 44th Int. International Conference Parallel Processing*, Beijing, China, 2015, pp. 629–638.

[4] L. Tan, S. L. Song, P. Wu, Z. Chen, R. Ge, and D. J. Kerbyson, Investigating the interplay between energy efficiency and resilience in high performance computing, in *Proc. 29th Int. Parallel and Distributed Processing Symposium*, Hyderabad, India, 2015, pp. 786–796.

[5] S. Rivoire, M. A. Shah, P. Ranganathan, and C. Kozyrakis, Joulesort: A balanced energy-efficiency benchmark, in *Proc. 26th Int. Special Interest Group On Management of Data*, Beijing, China, 2007, pp. 365–376.

[6] A. Rasmussen, G. Porter, M. Conley, H. V. Madhyastha, R. N. Mysore, A. Pucher, and A. Vahdat, Tritonsort: A balanced large-scale sorting system, in *Proc. 8th Int. Usenix Conference on Networked Systems Design & Implementation*, Boston, MA, USA, 2011, pp. 1–28.

[7] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan, Fawn: A fast array of wimpy nodes, in *Proc. 22nd Int. Acm Symposium on Operating Systems Principles*, Montana, MT, USA, 2009, pp. 1–14.

[8] A. Tiwari, M. Schulz, and L. Carrington, Predicting optimal power allocation for cpu and dram domains, in *Proc. 29th Int. Parallel and Distributed Processing Symposium Workshop (IPDPSW)*, Hyderabad, India, 2015, pp. 951–959.

[9] H. Zhang and H. Hoffmann, Maximizing performance under a power cap: A comparison of hardware, software, and hybrid techniques, *ACM SIGPLAN Notices*, vol. 51, no. 4, pp. 545–559, 2016.

[10] R. Ge, X. Feng, Y. He, and P. Zou, The case for cross-component power coordination on power bounded systems, in *Proc. 45th Int. International Conference on Parallel Processing (ICPP)*, Philadelphia, PA, USA, 2016, pp. 516–525.

[11] M. Chen, X. Wang, and X. Li, Coordinating processor and main memory for efficientserver power control, in *Proc. 25th Int. International Conference on Supercomputing (ICS)*, Arizona, AZ, USA, pp. 130–140.

[12] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini, CoScale: Coordinating CPU and memory system DVFS in server systems, in *Proc. 45th Int. International Symposium on Microarchitecture (MICRO)*, Canada, 2012, pp. 143–154.

[13] J. Rubio, K. Rajamani, F. Rawson, H. Hanson, S. Ghiasi, and T. Keller, Dynamic processor overclocking for improving performance of power-constrained systems, Report, IBM, 2005.

[14] A. D. M. Akhshabi1, Overclocking of CPU and graphics cards cooling refrigerator models offer the xtreme (permanent use) in order to increase efficiency, *Bulletin of Applied and Research Science*, vol. 3, no. 3, pp. 44–50, 2013.

[15] C. Bienia, S. Kumar, J. P. Singh, and K. Li, The parsec benchmark suite: Characterization and architectural implications, in *Proc. 17th Int. International Conference on Parallel Architectures and Compilation Techniques*, Raleigh, NC, USA, 2008, pp. 72–81.

[16] P. R. Luszczek, D. H. Bailey, J. J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, and D. Takahashi, The HPC challenge (HPCC) benchmark suite, in *Proc. 19th Int. ACM/IEEE Conference on Supercomputing*, Tampa, SF, USA, 2006, pp. 213–213.

[17] Intel 64 and IA-32 Architectures Software Developers Manual, Intel Corporation, 2014.

[18] D. James, How to overclock: It's easier than you think, https://www.pcgamesn.com/hardware-guides/overclocking-guide-how-to-overclock, 2017.

[19] S. Moment, DDR4 RAM overclocking 101 guide, http://www.overclockers.com/forums/showthread.php/785102-DDR4-RAM-overclocking-101-guide, 2017.

[20] D. Lo and C. Kozyrakis, Dynamic management of turbomode in modern multi-core chips, in *Proc. 20th Int. High Performance Computer Architecture (HPCA)*, Florida, FL, USA, 2014, pp. 603–613.

[21] Intel vtune amplifier, https://software.intel.com/en-us/intel-vtune-amplifier-xe, 2017.
[22] M. Dimitrov, Intel power governor, https://software.intel.com/en-us/articles/intel-power-governor, 2012.
[23] V. Viswanathan, Intel Memory Latency Checker v3.4, https://software.intel.com/en-us/articles/intelr-memory-latency-checker, 2017.
[24] C. Lefurgy, X. Wang, and M. Ware, Power capping: A prelude to power shifting, *Cluster Computing*, vol. 11, no. 2, pp. 183–195, 2008.
[25] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, No power struggles: Coordinated multi-level power management for the data center, in *Proc. 13rd Int. International Conference on Architectural Support for Programming Languages and Operating Systems*, Seattle, WA, USA, 2008, pp. 48–59.
[26] X. Yang, Y. Zhang, X. Lu, J. Xue, I. Rogers, G. Li, G. Wang, and X. Fang, Exploiting the reuse supplied by loop-dependent stream references for stream processors, *ACM Transactions on Architecture and Code Optimization*, vol. 7, no. 2, pp. 1–35, 2010.
[27] X. Yang, Z. Wang, J. Xue, and Y. Zhou, The reliability wall for exascale supercomputing, *IEEE Transactions on Computers*, vol. 61, no. 6, pp. 767–779, 2012.
[28] B. Rountree, D. K. Lownenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, and T. Bletsch, Adagio: Making DVS practical for complex HPC applications, in *Proc. 23rd Int. International Conference on Supercomputing*, Yorktown Heights, NY, USA, 2009, pp. 460–469.
[29] S. Bhalachandra, A. Porterfield, S. L. Olivier, and J. F. Prins, An adaptive core-specific runtime for energy efficiency, in *Proc. 31s Int. IEEE International Parallel and Distributed Processing Symposium*, Florida, FL, USA, 2017, pp. 947–956.
[30] A. Marathe, P. E. Bailey, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, A run-time system for power-constrained hpc applications, in *Proc. 31s Int. High Performance Computing*, Bengaluru, Indian, 2015, pp. 394–408.
[31] I. Stamelakos, S. Xydis, G. Palermo, and C. Silvano, Variation-aware voltage island formation for power efficient near-threshold manycore architectures, in *Proc. 19th Int. Asia and South Pacific Design Automation Conference*, Singapore, 2014, pp. 304–310.
[32] U. R. Karpuzcu, A. Sinkar, N. S. Kim, and J. Torrellas, Energysmart: Toward energy-efficient manycores for near-threshold computing, in *Proc. 19th Int. High Performance Computer Architecture*, Shenzhen, China, 2013, pp. 542–553.
[33] R. Begum, D. Werner, M. Hempstead, G. Prasad, and G. Challen, Energy-performance trade-offs on energy-constrained devices with multi-component DVFS, in *Proc. 10th Int. International Symposium on Workload Characterization*, Georgia, GA, USA, 2015, pp. 34–43.
[34] S. Mittal, A survey of architectural techniques for DRAM power management, *International Journal of High Performance Systems Architecture*, vol. 4, no. 2, pp. 110–119, 2012.
[35] Q. Liu, M. Moreto, J. Abella, F. J. Cazorla, and M. Valero, Dream: Per-task DRAM energy metering in multicore systems, in *Proc. 20th Int. European Conference on Parallel Processing*, Porto, Portugal, 2014, pp. 111–123.
[36] Q. Deng, Active low-power modes for main memory with memscale, *IEEE Micro*, vol. 32, no. 3, pp. 62–69, 2012.
[37] P. Zou, T. Allen, C. H. Davis IV, X. Feng, and R. Ge, Clip: Cluster-level intelligent power coordination for power-bounded systems, in *Proc. 20th Int. Cluster Computing*, Hawaii, HI, USA, 2017, pp. 541–551.
[38] R. Ge, P. Zou, and X. Feng, Application-aware power coordination on power bounded NUMA multicore systems, in *Proc. 46th Int. International Conference on Parallel Processing*, Briston, UK, 2017, pp. 591–600.
[39] B. Acun and L. V. Kale, Mitigating processor variation through dynamic load balancings, in *Proc. 30th Int. International Parallel and Distributed Processing Symposium Workshops*, Chicago, IL, USA, 2016, pp. 1073–1076.
[40] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, Exploring hardware overprovisioning in power-constrained, high performance computing, in *Proc. 27th Int. International Conference on Supercomputing*, Eugene, OR, USA, 2013, pp. 173–182.

Feihao Wu received the BS degree from Harbin Institute of Technology, China, and now is a master student at National University of Defense Technology. His research interests include the large scale parallel numerical simulation and energy efficiency computing.

Juan Chen received the PhD degree from National University of Defense Technology, China. She is now an associate professor at National University of Defense Technology, China. Her research interests include supercomputer systems and energy-aware interconnection network design.

Yong Dong received the PhD degree from National University of Defense Technology, China. He is now an associate professor at National University of Defense Technology, China. His main research interests include supercomputer systems and storage systems.

Wenxu Zheng received the BS degree from National University of Defense Technology, China, and now is a master student at National University of Defense Technology. His main research interest is the communication in large scale parallel computing.

Xiaodong Pan received the BS degree from Wuhan University of Technology, China, and now is a master student at National University of Defense Technology. His main research interest is the communication in large scale parallel computing.

Yuan Yuan received the PhD degree from National University of Defense Technology, China. He is now an associate professor at National University of Defense Technology, China. His research interests include supercomputer systems and HPC monitoring and diagnosis.

Zhixin Ou now is an undergraduate student at National University of Defense Technology. Her main research interest is the energy efficiency of super computer.

Yuyang Sun now is an undergraduate student at National University of Defense Technology. His main research interest is the power assumption in super computer.