Energy Efficient Proactive Thermal Management in Memory Subsystem *

Raid Ayoub
rayoub@cs.ucsd.edu

Krishnam Raju Indukuri
kindukur@ucsd.edu

Tajana Simunic Rosing
tajana@ucsd.edu

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404

ABSTRACT

Energy management of memory subsystem is challenging due to performance and thermal constraints. Big energy gains can be obtained by clustering memory accesses, however this also leads to a higher need for cooling due to larger temperatures in active areas of memory. Our solution to memory thermal management problem is based on proactive thermal management that intelligently allocates workload pages to few memory units and powers down rest of the memory. Our experimental results show that this approach improves energy savings by 43% and reduces performance overhead by 85% with respect to the state of the art polices.

Categories and Subject Descriptors

B.3 [**MEMORY STRUCTURES**]: B.3.1 Semiconductor Memories

General Terms

Management, Design, Reliability, Performance

Keywords

Memory subsystem, Thermal management, Proactive, Energy, Performance

1. INTRODUCTION

State of the art server systems are equipped with multiple CPU sockets where each socket contains multiple cores. To cope with the substantial memory demand from these processors, density and speed of memory DRAM subsystem have been increased dramatically. However, such increase in memory size and speed have led to a significant increase in total energy consumed by the memory. In addition, DRAM subsystem has become a thermal hot spot due to its high power density. Managing DRAM temperature is of paramount importance due to the fact that reliability decreases exponentially with temperature [14]. High-end-servers are usually equipped with fan subsystems that use forced convection to expel the extra heat. However, fan subsystem itself is a power hungry module which consumes a big portion of the total system energy [9]. The cost of cooling is expected to escalate at rates higher than the growth in power density due to the cubic relationship between fan power and its speed [13]. Recent studies show a growing concern for energy problems in memory subsystem and that the memory energy may even surpass energy consumption of the CPU, which is traditionally known to be a primary source for energy consumption in computer systems [3, 6].

Memory subsystem consumes around (20-30)% of its total power when idle or close to idle. Memory, in a way, wastes power in this state because it does no efficient work. A simple solution is to consolidate memory references to a subset of memory DIMMs (dual in-line memory module) and put rest of the DIMMs in deep sleep mode that consumes about 1% of total energy. Although this solution seems obvious, there are big challenges in adopting it. The major problem is to handle the expected increase in thermal emergencies which impact cooling energy and performance overhead.

In this paper we propose a new technique that delivers appreciable energy savings in memory sub-system and minimize thermal problems. We adopt the consolidation method, as it is a natural way to approach this problem. Our main contribution is in managing potential thermal emergencies that may result from consolidation while still providing decent energy savings. To do this, we perform intelligent migration of memory accesses between the active and dormant DIMMs while minimizing performance overhead. We show in this work that migrating a small portion of memory pages (active pages) is sufficient since a large portion of the total pages are typically dormant. Our profiling analysis shows that the inherent access pattern in these pages allow us to capture them within a small window of time to enable quick migration. To avoid thermal emergencies, we use a proactive approach where we initiate the migration process early enough to allow for sufficient number of active pages to be migrated. This also helps lower power density of hot DIMMs. To predict thermal emergencies, we developed a novel temperature predictor that requires no runtime adaptations and can be employed at negligible cost. The experimental results we report show that our approach improves

*This work has been funded by NSF Project GreenLight grant 0821155, NSF SHF grant 0916127, MuSyC, UC Micro grant 08-039, Sun Microsystems, and Cisco.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISLPED'10, August 18–20, 2010, Austin, Texas, USA.

Copyright 2010 ACM 978-1-4503-0146-6/10/08 ...$10.00.

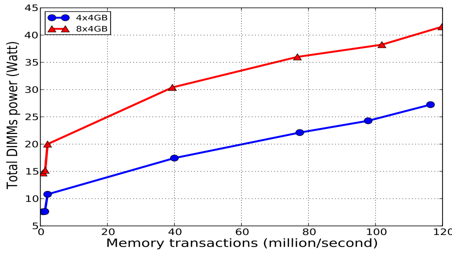

Figure 1: Total power of 8 and 4 DIMMs

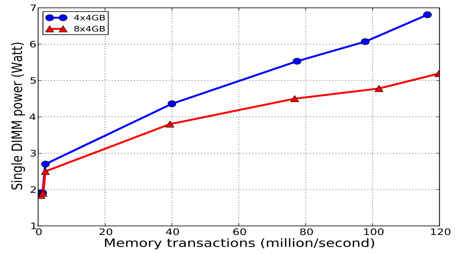

Figure 2: Single DIMM power of 8 and 4 DIMMs

energy savings by 43% and reduces performance overhead by 85% with respect to the state of the art polices.

2. RELATED WORK

The research in recent years has sought a number of techniques to handle the energy problem in memory. The work in [7] proposed a technique to lower DRAM power by changing memory mode to low power when there are no accesses to the memory. This approach is beneficial only when there are frequent idle periods. Research in [8] manages memory power by clustering heavily accessed pages to a subset of the DIMMs and put the rest of the DIMMs in a low power mode to save power. However, such consolidation is likely to generate thermal problems. In all these techniques, the thermal issues are not examined.

To handle thermal problems in memory subsystem, a few techniques have been suggested [11, 10]. The work in [11] mitigates overheating in memory system by adjusting memory throughput to stay below the emergency level. The primary drawback in this approach is the associated performance overhead. The work in [10] manages memory overheating by grouping threads in the system (both active and idle) in a way that each group is mapped to certain DIMMs assuming that all the threads in a group can be active simultaneously. Thus, only one group of DIMMs is active at any point in time while the rest stay inactive to cool them down until they become active again. A number of assumptions are made in this work, which can degrade its applicability. For example, the authors assume that bandwidth of threads is known in advance through static profiling. Additionally, this approach is restricted to the cases when number of threads is a multiple of number of cores in the system that may not suit many applications. It should be noted that none of these techniques consider cooling energy costs which can be significant.

To the best of our knowledge, there is no prior work that manages thermal, power and cooling costs of memory subsystems. The contributions of our work can be summarized as: 1) We propose a new proactive thermal management technique for memory subsystem that intelligently schedules pages of running applications across memory DIMMs. 2) We developed a new temperature predictor that is highly accurate, cost efficient and requires no adjustment to workload changes. The accuracy of the prediction is below 0.5^oC. In result, we get high energy savings with better performance.

3. MOTIVATION

We start with a brief background about memory systems. Typical DRAM subsystem is organized as an array of multiple DIMMs, where each DIMM is composed of dual ranks and each rank contains multiple memory chips (e.g. 8 chips per rank). The memory access commands are first sent to *memory controller*. The memory controller communicates with the DRAM subsystem through memory channels where each subgroup of the DIMMs is assigned to a single channel. As each channel can support only a limited bandwidth, the memory controller employs interleaving policy across DIMMs to maximize the use of all channels bandwidth by evenly distributing memory accesses across these channels and DIMMs.

DRAM power is a function of given memory state and can be controlled at the DIMM level as well as at the rank level. In general there are four primary states in DRAM: active, precharge, power-down, and self-refresh. Active state consumes the highest power whereas self-refresh consumes the lowest (around 1% of active standby power [8]). The time to wake DRAM up from self-refresh mode is in the range of 100s of CPU cycles. On the other hand power-down state consumes around 10% of the active state power with wake up time of few nanoseconds.

The DRAM energy consumed to do the actual job of retrieving or storing data is only a portion of the total energy. This is because DIMMs consume energy even when they are idle to maintain the stored information. To save energy we need to minimize the waste power that is not being used for serving requests. In order to do that, we can consolidate memory pages of active threads into a smaller set of DIMMs and place the rest in a low power mode.

To evaluate the benefits of consolidation we collected real life power data from actual DDR2 memory placed in a state of the art dual socket Xeon E5440 server. To estimate DIMM power we inserted current sensors in power supply lines of each DIMM and aggregated the data using data acquisition system. This system supports four memory channels where each is connected to a subset of DIMMs. In these experiments we use DIMMs of size 4GB each. To estimate the amount of memory accesses we use performance counters of Xeon processor and use *perfmon2* tool to access these counters. Figure 1 shows the total memory power for two memory configurations. In the first configuration, we use 4 DIMMs where each is connected to a separate memory channel to maximize bandwidth. For the second case, we use 8 DIMMs with 2 DIMMs per channel. The x-axis corresponds to memory transaction rate where each transaction is of L2 cache block size. Figure 1 clearly shows that a significant savings of more than 40% can be achieved with consolidation. More savings are expected if a larger set of DIMMs is used for the base line case.To minimize the impact on performance we allow interleaving within and across the set of active DIMMs. From the real measurements we observed that the associated performance loss is quite small (less than 1%) which could be related to the bank conflicts

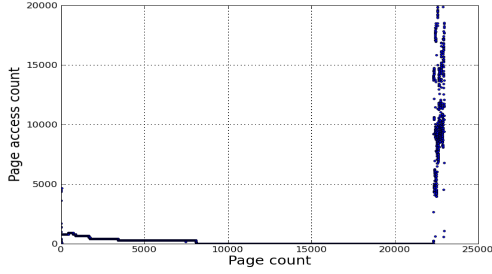

Figure 3: Page access patterns of memory intensive application, mcf

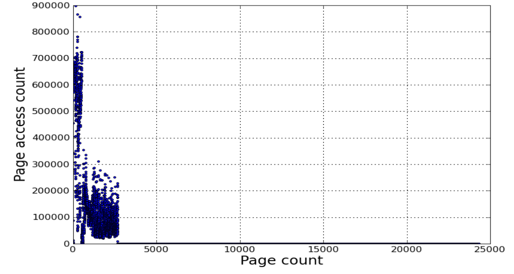

Figure 4: Page access patterns of CPU intensive application, bzip2

within the DIMMs. Figure 2 shows the impact of consolidation on power per DIMM. It can be seen that power density can increase by 25% which will cause temperature problems that must be addressed at a minimal overhead. In the following sections we elaborate on how to reduce potential thermal problems while still maintaining appreciable energy savings.

4. THERMAL AWARE PAGE MANAGEMENT

In this section we discuss feasible solutions for managing potential heat problems that may occur as a result of workload consolidation in DIMMs. The simplest solution is to throttle memory accesses when temperature reaches a thermal emergency. However, such approach may lead to a large performance overhead. One other solution is to increase fan speed to cool the DIMMs. However, as fan speed is closely related to fan power, this may lead to an increase in net energy consumption. The solution that we propose is to migrate memory accesses to colder DIMMs as the temperature on the working set approaches threshold and place the hotter ones in a self-refresh mode. Further, the average temperature is expected to reduce due to an overall reduction of idle power, which in turn lowers the chances of thermal emergencies.

A number of challenges need to be addressed. During migration, both hot and cold DIMMs must stay active and consume energy, so a swift migration mechanism is necessary to minimize idle power. We also need to address the initiation time for the migration. If we react after the temperature reaches threshold, the benefit of migration may be limited especially when temperature rise is steeper than the rate at which heat is removed via migration. This scenario is likely to happen if migration rate is low or if we are migrating dormant pages. The speed of migration is limited by how many pages we can migrate in a given period of time. This limitation is a function of memory bus bandwidth and ratio of execution to migration time. This ratio needs to be high to maintain responsiveness. These issues can be resolved if we can predict thermal emergencies and start acting ahead of time. In summary, a robust thermal management system can be built through proactive approach that focuses on migrating just what is needed.

4.1 Exploitation of page clustering

To address migration efficiency, we first need to understand the characteristics of access patterns of memory pages. Figures 3 and 4 show the page access patterns of memory and cpu intensive applications respectively. We use m5, micro-architectural simulator, to generate these statistics [4]. From these results, it can be seen that the number of active pages is significantly smaller than the total number of pages in an application. Focusing on migrating active pages is the major premise for implementing a low cost migration. For this approach to be applicable we need to capture these pages within a short window of migration. We found that it is common for applications to read or write a set of active pages before a given page gets referenced again. This means that the total set of active pages can be captured in a relatively short interval (e.g. few milliseconds or less), which is more than sufficient to implement a fast migration.

4.2 Temperature modeling and prediction

Predicting temperatures for long enough time allows a system to react swiftly and avoid thermal emergencies. To meet the migration constraints that we just discussed, we may need to predict temperature pattern hundreds of milliseconds into the future (e.g 400ms for migrating 1000 pages at a rate of 25 pages per 10ms). The time constant of the DRAM die is typically much smaller than this range (only a few 10s of milliseconds). However, we show that DIMM temperature is mainly determined by their thermal package time constant which is much larger than the die time constant (order of 10s of seconds when using a usual heat spreader). Large time constant of thermal package is the key idea for a design of thermal predictor with enough prediction span.

A DIMM is usually composed of dual ranks where each rank contains a number of DRAM chips. Each DRAM chip is designed to store a portion of data. As a result, DIMM power is distributed almost equally across DRAM chips. However, the temperature across DIMMs may not be uniformly distributed mainly because of the increase in airflow temperature as it passes along the DIMM, which could make the later DRAM chips hotter than the ones in the front. The DRAM chips also have poor heat transfer with the ambient which results in higher temperature. To improve thermal distribution and heat transfer to the ambient, DIMMs are usually equipped with heat spreaders. Figure 5 shows the thermal model of a DIMM that uses a heat spreader. This model is based on the known duality between electrical current and heat [15]. In this figure, P_{chip} represents the power dissipated in each DRAM chip, R_{chip} is the vertical thermal resistance of each chip, C_{chip} is the thermal capacitance of each chip and T_j is the junction temperature of a DRAM chip. For the heat spreader, R_{sp} and C_{sp} represent its thermal resistance and capacitance respectively. The heat transfer between heat spreader and ambient is modeled by a convective resistor, R_{conv}. The heat spreader is assumed as a single node due its small horizontal thermal resistance compared to the convective resistance. To understand the ef-

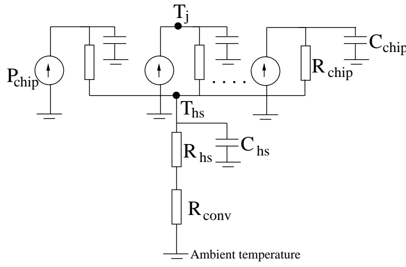

Figure 5: DIMM thermal model

fect of thermal package (heat spreader) on the temperature breakdown, let's start with computing the steady-state value of T_j which can be calculated as:

$$T_j = P_{chip}R_{chip} + T_{hs} \quad (1)$$

where T_{hs} is the heat spreader temperature on the side that is in contact with DRAM chips. This temperature can be computed as:

$$T_{hs} = NP_{chip}(R_{hs} + R_{conv}) \quad (2)$$

Where N is the number of DRAM chips. The values of R_{chip} and R_{conv} are typically comparable [1, 11]. As a result it can be seen that the effect of heat spreader temperature on T_j breakdown is about N times the local heat generated within the DRAM chip. Based on this analysis, one can expect that the transient temperature of T_{hs} would also dominate the value of T_j. Time constant of heat spreader is in the order of 10s of seconds [11]. This indicates that the DRAM temperature would change at a rate similar to that of heat spreader.

Temperature prediction: To predict the DRAM temperature far ahead in time (100s of milliseconds), we use the fact that temperature dynamics of T_j depend largely on temperature dynamics of heat spreader. The instantaneous temperature of heat spreader can be estimated based on the RC network model of heat spreader as follows:

$$T_{hs}(t) = T_{ss}(1 - e^{-t/\tau_{hs}}) + T_o e^{-t/\tau_{hs}} \quad (3)$$

where τ_{hs} is the heat spreader time constant, $\tau_{hs} = (R_{hs} + R_{conv})C_{hs}$, T_{ss} is the steady state temperature for a given DRAM power, and T_o is the initial temperature. To predict the future temperature, we use derivative analysis to extract a recursive formula, which can be computed at negligible overhead. The first derivative of (3) is shown in the following:

$$\frac{dT_{hs}(t)}{dt} = \frac{1}{\tau_{hs}}(T_{ss} - T_o)e^{-t/\tau_{hs}} \quad (4)$$

Using this equation, we can write a recursive form for the derivative of T_{hs} at the prediction point $t + \delta t$ as:

$$\frac{dT_{hs}(t + \delta t)}{dt} = \frac{dT_{hs}(t)}{dt} e^{-\delta t/\tau_{hs}} \quad (5)$$

Given (5), we can write the recursive formula for predicting change in temperature, $\Delta T_{hs}(t + \delta t)$:

$$\Delta T_{hs}(t + \delta t) = \Delta T_{hs}(t)e^{-\delta t/\tau_{hs}} \quad (6)$$

To get the final formula for predicting T_j, we exploit the big difference between the time constant of heat spreader and that of die. This indicates that die temperature will be in steady-state condition while heat spreader temperature is still changing towards its steady-state value. Putting all this together, the future temperature of T_j can be computed easily as:

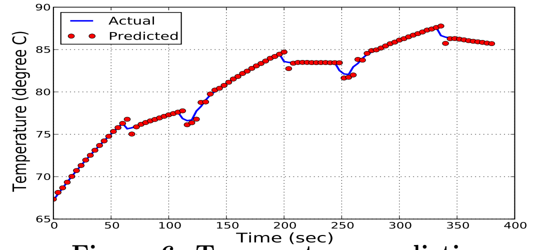

Figure 6: Temperature prediction

$$T_j(t + \delta t) = T_j(t) + \Delta T_{hs}(t + \delta t) \quad (7)$$

In Figure 6 we show an illustrative example of applying our predictor with prediction distance of 4 seconds. In this example we run a sequence of memory intensive applications (3 instances of *equake*) followed by CPU intensive one (one instance of *perl*) as follows (3*equake*, 1*perl*, 3*equake*, 1*perl*, 3*equake*). We run these benchmarks on a Xeon server and collect power traces using the setup described in section 3. To get temperature results, we extended hot-spot simulator [15] with a DRAM thermal model. To ensure accuracy, we measured the value of convective resistance from our real test bed and also included the extra heat that is generated by CPU since DRAM sub-system is placed after CPU sockets in this particular server. In this example we assume ambient temperature of 40^oC with no thermal management and fan spinning at idle speed. It is clear from the results that our predictor is highly accurate, with an average error of about 0.15^oC. The prediction overhead is minuscule at less than a few CPU cycles, as generating new prediction requires evaluating two very simple equations (6) and (7).

4.3 Thermal management polices

The primary objective of this work is to minimize energy of memory subsystem while respecting system performance. Before we introduce our policy, we elaborate on few state of the art thermal management polices that we compare our work to.

State of the art systems manage thermal emergencies by increasing the fan speed, gating the memory accesses or both. However, these techniques induce energy and performance overhead. We implement two related polices. The first policy uses memory gating where memory accesses are prevented for a few microseconds when temperature threshold is reached, referred to as ***Gating***. In the second policy we include the effect of fan in expelling extra heat and if it proves insufficient we use Gating policy as a back up. We denote this policy, ***Fan + Gating*** [11]. We also implemented the following more sophisticated policies:

Reactive Page Allocation, RPA: In this policy we use DIMM consolidation concept to save energy. At any point in time, we activate one group of DIMMs where no two DIMMs can belong to the same memory channel to maximize memory bandwidth. The rest of the DIMMs are placed in a self-refresh mode to save energy. When the workload requires more memory, additional DIMMs can be activated. When thermal threshold is reached, we start migrating all pages that belong to a workload to the coolest set of DIMMs. After the migration is completed we place the source DIMMs in a self-refresh mode. To avoid possible ping-ponging, the migration is prevented unless there is enough temperature difference between the hot and cold DIMMs. We choose this temperature difference at 4^oc which translates into a period

of 50-100s between any two migrations. During migration, we employ the *Gating* policy for thermal emergency.

Proactive Page Allocation, PPA: This policy is the main contribution of this work. We use DIMM consolidation concept to save energy. We choose a set of active DIMMs in a way similar to that of RPA. The rest of the DIMMs are placed in a self-refresh mode to save energy. This policy uses our predictor to determine the thermal emergencies ahead of time. Upon predicting thermal emergencies, we exploit the page clustering concept and start migrating only the active pages. As active pages are typically clustered, we can use a time-out policy of 10ms to determine when they have been migrated. If the timeout threshold is exceeded and no more pages are accessed, we place the hot DIMMs in self-refresh mode. We avoid possible ping-ponging using temperature difference methods similar to what use in RPA. During migration, we employ the *Gating* policy to manage any thermal emergencies.

5. EVALUATION

5.1 Methodology

To evaluate our algorithm we use a state of the art system that consists of 45nm Intel Quad Core dual socket Xeon E5440 servers and DDR2 memory subsystem. In our system there are 4 memory channels where each is associated to a number of DIMMs (maximum 4 DIMMs per channel). A set of two channels is connected to a single memory controller and can sustain a bandwidth of 8.4GB/sec. To measure actual power of each DIMM we inserted current sensors in the power path of DIMM. These power values are then used to estimate temperature using our extended version HotSpot simulator. For the thermal model parameters, we obtained the value of R_{chip}, $4^oC/W$, from micron SPEC sheet[1]. We extracted the value of $R_{hs} + R_{convec}$, and C_{hs} using our model and real life temperature data. We used the built-in thermal sensors in DRAM chips to collect temperature data using IPMI (Intelligent Platform Management Interface) tools. To model the effect of airflow we measured the value of $R_{hs} + R_{convec}$ at different levels of airflow. In our test bed there are two fans associated with each socket that can provide max airflow of 53.4 CFM at max power of 29.4 Watts [2]. The airflow first cools CPU sockets and then passes to memory subsystem. To consider the effect of CPU in heating up the airflow, we measured air temperature at the DIMMs. For the fan control algorithm we assumed a closed loop controller similar to one used in modern systems [5].

For memory configuration we use two combinations. This first one has 8 DIMMs (total of 32GB) to resemble the base line case (a pair of DIMMs are connected to one memory channel to maximize bandwidth). For the consolidation case we place half of the DIMMs in self-refresh mode where each active DIMM is connected to one memory channel. To capture active pages at run time we use $PEBS$, a hardware support that stores physical address of active pages in a certain place in memory to allow operating system to read this information. Our algorithm is deactivated if memory footprint of a workload surpasses the size of consolidated memory or if there is congestion in memory channels. During migration, we choose to migrate a maximum of 25 pages every 10 ms. We assume the critical temperature as 85 ^oC which is common to such designs [10] and ambient temperature of 42 ^oC. For our prediction policy, we use a prediction span of 4^oC that is sufficient to avoid possible overheating.

Table 1: Benchmarks characteristics

Benchmark	Power per DIMM (Watt)	Active pages	Total pages
bzip2	1.8	1015	24600
perl	1.9	180	200
gcc	2.5	3500	18150
mcf	3.8	2990	25000
equake	4.3	5200	6200

We use benchmarks from the SPEC2000 suite for our workloads . We selected a set of benchmarks that exhibit various levels of memory access rates to represent real life applications (see Table 1). The second column shows the power dissipation caused by workloads in each DIMM in the case when all 8 DIMMs are active. The third and fourth column give number of active pages and total pages in the application. To better evaluate our policies under various workload conditions, we select benchmark combinations that have different average power intensity and variance. We run mulitple instances of each combination to generate a big enough window for our test case.

5.2 Results

The reported results show that our prediction policy outperforms other policies in improving both net performance and energy savings. Figure 7 depicts the performance overhead associated with the policies: Gating:8DIMMs (8 active DIMMs), Fan+Gating:4DIMMs, Gating:4DIMMs (4 active DIMMs), RPA and PPA. It is evident from the results that PPA gives a sizable improvement over Gating:8DIMMs, Fan+Gating:4DIMMs, Gating:4DIMMs, RPA with an average performance overhead reduction of 86%, 15%, 93% and 59% respectively. The PPA policy has better performance relative to Gating:8DIMMs despite the fact that the latter distributes memory accesses evenly across all DIMMs. These savings stem from reducing the amount of idle power which in turn decreases average temperature and the frequency of thermal emergencies. Results of executing the policies on workload *3mcf+3gcc* show that the performance overhead with PPA is negligible compared to Gating:8DIMMs. The savings for the workload, *4equake+2bzip2* is slightly reduced because of its very high power density. The overhead in the case of *4mcf+2perl* is higher than the case of *4mcf* across all polices. This is because, *perl* is highly CPU intensive and generates extra heat within CPU.

We evaluate Fan+Gating:4DIMMs policy which does naive consolidation but benefits from fan cooling to minimize hot-

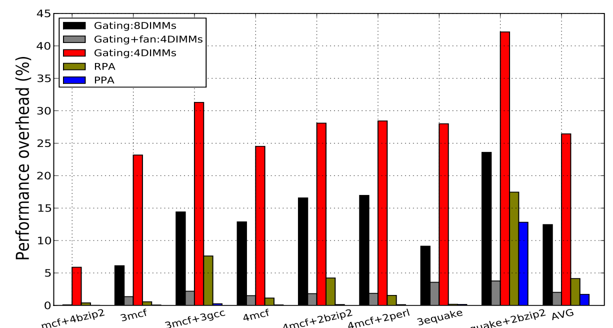

Figure 7: Performance overhead

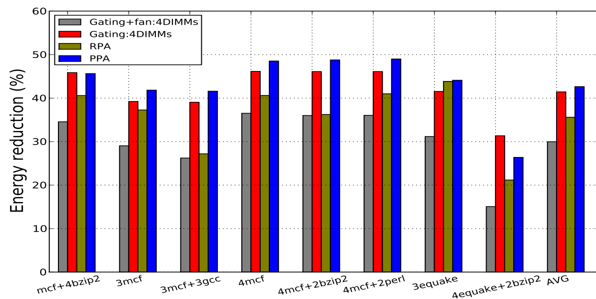

Figure 8: Energy savings

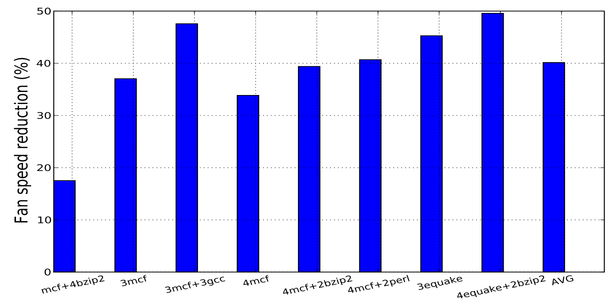

Figure 9: Fan speed reduction using PPA

spots. Although the performance is improved with this policy, fan consumes considerable energy which makes it less attractive as shown in Figure 8. Fan system is unable to eliminate all thermal emergencies because of its slower response time. The results also show that PPA gives better performance overhead reduction compared to RPA for all the workloads. This is because PPA acts ahead of time by predicting thermal emergencies and also employs intelligent page migration. Performance overhead of RPA increases significantly for 4mcf+2bzip2 compared to 4mcf alone since most of the pages in 2bzip2 are dormant. It can be noticed that this overhead remains negligible for PPA in both the cases.

We plotted energy savings of all policies with respect to Gating:8DIMMs policy in Figure 8. It is evident from these results that PPA significantly outperforms Gating:8DIMMs, with an average improvement of 43%. This is because PPA brings a substantial reduction in the idle power consumption. The PPA far surpasses Fan+Gating:4DIMMs due to the extra energy consumed by the fan subsystem in the latter policy. This margin becomes larger as thermal stress increases due to the cubic dependency between fan speed and its power. Though the results show that Gating:4DIMMs delivers energy savings comparable to PPA, it suffers from a high performance overhead as discussed in Figure 7. The PPA outperforms RPA by 15% in the case of *3mcf+3gcc* because RPA ends up using much higher migration window. For the case of *4equake+2bzip2*, the energy savings of PPA mainly come form predicting and managing thermal problems ahead of time since this workload induces high temperatures.

Figure 9 shows the reduction in fan speed using PPA with respect to Fan+Gating:4DIMMs. The savings are appreciable, with an average improvement of 40%. The significance of this improvement lies not only in saving net energy but also in maximizing the fan lifetime and minimizing acoustic noise [12]. Further, PPA brings about an 8.3% reduction in the average number of thermal emergencies compared to Fan+Gating:4DIMMs. This clearly indicates that PPA, in addition to reducing the fan speed significantly, improves the overall thermal behavior of the memory system. In summary, PPA is highly effective in improving energy savings and reducing thermal problems in memory subsystem.

6. CONCLUSION

In this work we have proposed a proactive scheduling technique to minimize energy consumption in memory subsystem. Our algorithm incorporates continuous temperature measurement and memory page characterization to guide page allocation and DIMMs' power state configuration in a thermally aware process. To save energy we consolidate memory references to a smaller set of DIMMs and thus reduce the impact of idle power. The challenge we address lies in finding a novel technique that can reduce thermal stress while maximizing performance. To accomplish this, we use a common page clustering behavior and develop a temperature prediction technique. Our analysis show that our predictor is highly accurate. Its parameters can be extracted at design time and the prediction can be computed in a few CPU cycles. Our results show that this technique improves energy savings by 43% and decreases performance overhead by 85% with respect to state of the art polices.

7. REFERENCES

[1] *www.micron.com/products/dram/.*

[2] *www.sunon.com.*

[3] L. Barroso and U. Holzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[4] N. Binkert, R. Dreslinski, L. Hsu, K. Lim, A. Saidi, and S. Reinhardt. The m5 simulator: Modeling networked systems. *IEEE Micro*, 26(4):52–60, 2006.

[5] H. Chiueh, L. Luh, J. Draper, and J. Choma. A novel fully integrated fan controller for advanced computer systems. *SSMSD*, pages 191–194, 2000.

[6] G. Dhiman, R. Ayoub, and T. Rosing. Pdram: a hybrid pram and dram main memory system. *DAC*, pages 664–469, 2009.

[7] X. Fan, C. Ellis, and A. Lebeck. Memory controller policies for dram power management. *ISLPED*, pages 129 – 134, 2001.

[8] H. Hai, S. Kang, L. Charles, and K. Tom. Improving energy efficiency by making dram less randomly accessed. *ISLPED*, pages 393–398, 2005.

[9] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. Keller. Energy management for commercial servers. *IEEE Computer*, pages 39–48, 2003.

[10] C.-H. Lin, C.-L. Yang, and K.-J. King. Ppt: Joint performance/power/thermal management of dram memory for multi-core systems. *ISLPED*, pages 93–98, 2009.

[11] J. Lin, H. Zheng, Z. Zhu, H. David, and Z. Zhang. Thermal modeling and management of dram memory systems. *ISCA*, pages 312–322, 2007.

[12] G. Paparrizos. An integrated fan speed control solution can lower system costs, reduce acoustic noise, power consumption and enhance systemreliability. *Technical report, Microchip Technology Inc*, 2003.

[13] M. Patterson. The effect of data center temperature on energy efficiency. *Proc. ITHERM*, pages 1167–1174, 2008.

[14] M. Pedram and S. Nazarian. Thermal modeling, analysis, and management in vlsi circuits: principles and methods. *Proceedings of the IEEE*, pages 1487–1501, 2006.

[15] K. Skadron, M. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *TACO*, pages 94–125, 2004.