HyMAD: a Hybrid Memory-Aware DVFS strategy

Camélia Slimani
Univ. Bretagne Occidentale,
UMR6285, Lab-STICC, France
Ecole Sup. Info., Algeria
camelia.slimani@univ-brest.fr

Stéphane Rubini
Univ. Bretagne Occidentale,
UMR6285, Lab-STICC, France
rubini@univ-brest.fr

Jalil Boukhobza
Univ. Bretagne Occidentale,
UMR6285, Lab-STICC, France
boukhobza@univ-brest.fr

ABSTRACT

Non-volatile memories, such as Phase Change Memories (PCM), have interesting energy properties. In effect, their static energy consumption is negligible while the consumed dynamic energy depends on the performed operation (read-/write). Several Dynamic Voltage and Frequency Scaling (DVFS) mechanisms have been proposed to optimize the energy consumption of memory-bound tasks in embedded systems. In a hybrid memory, using both DRAM and PCM, these DVFS strategies need to be adapted to take into account the different energy behaviors of both memories. In this paper, we propose a Hybrid Memory-Aware DVFS strategy (HyMAD) to reduce the energy consumption for memory bound tasks. This mechanism relies on both the rate of PCM write operations and the overall memory access rate to tune the CPU frequency. HyMAD takes also into account the priority of a task to tune the "performance loss" / "energy efficiency" trade-off. We compared HyMAD with a state-of-the-art technique by evaluating the Energy-Squared-Delay product ($ED2P$). HyMAD $ED2P$ enhancement was evaluated between 2-45% as compared to a system without DVFS and up to 20% as compared to a state-of-the-art DVFS strategy.

CCS Concepts

•Hardware → *Non-Volatile Memory;* **Memory and dense storage; •Software and its engineering** → **Memory Management; •Operating Systems** → **performance;**

1. INTRODUCTION

Nowadays, the volume of data to process in embedded systems is growing exponentially [12]. Consequently, a higher main memory capacity is required to offer a satisfying computing capacity. However, DRAM technology integration has reached its limit in term of density, it can hardly scale with the growing application needs [11].

Emerging Non-Volatile Memories (NVM), such as PCM, have interesting properties and can be a solution to such an issue. In effect, PCM has a higher density than DRAM [10]. In addition, its non-volatility property makes the static energy very low as compared to DRAM that requires periodic refreshing. In terms of performance and dynamic energy consumption, PCM and DRAM are nearly equivalent for read operations. However, write operations on PCM are significantly slower and consume more energy as compared to DRAM [2]. In this context, it is important to design systems that take advantage of PCM properties whilst avoiding their energy overhead.

One of the most popular energy saving technique is the Dynamic Voltage and Frequency Scaling (DVFS). DVFS makes it possible to act on the CPU frequency to reduce the CPU dynamic power, and thus, the total system energy. Several state-of-the-art studies proposed to exploit the memory access boundedness of a task to reduce the CPU frequency [3, 6]. The basic idea behind these studies is that the more a task spends time in memory access, the more the frequency can be decreased. This allows to save energy without incurring a significant performance loss [3].

In a system equipped with a hybrid main memory composed of DRAM and PCM, having different energy properties, we believe that frequency scaling needs to be performed according to memory patterns.

In this paper, we propose a **Hy**brid **M**emory-**A**ware **D**VFS (HyMAD) technique that takes into account the heterogeneity of the memory. HyMAD relies on optimizing a performance loss/energy efficiency trade-off. We used Energy-Delay Product (EDP) and Energy Delay squared Product (ED^2P) to estimate this trade-off. These two metrics have been used in several DVFS studies [4]. The basic idea behind HyMAD is to build a frequency scaling model based on the energy properties of the system (CPU, DRAM and PCM) with the objective to minimize EDP/ED^2P metrics.

HyMAD is based on two main phases, an off-line phase in which the frequency scaling model is built based on the real system energy and performance properties (using local minimum search). CPU, DRAM and PCM are the main components accounted for. The second phase is an on-line phase during which the more adequate frequency is chosen according to the application profile using the built model.

We also considered the task static priority in frequency scaling. The Linux kernel uses this metric to compute the time slice duration of a task. The higher the static priority, the higher the time slice. Then, we found it irrelevant to reduce the frequency for tasks having a high static priority.

Our experiments show that HyMAD enhances ED^2P by

EWiLi'18, 4 October 2018, Torino, Italy.

2-45% as compared to no DVFS, and up to 20% as compared to state-of-the-art strategy [3].

This paper is organized as follows: Section 2 gives some background knowledge in addition to some related work summary. Section 3 describes HyMAD design. Section 4 discusses the evaluation methodology and results. We finally conclude and give some perspectives in Section 5.

2. BACKGROUND AND RELATED WORK

2.1 Background on memory architecture

Table 1 shows a comparison between PCM and DRAM established from different state-of-the-art surveys [2], [13], [5]. The advantages of PCM memories are shown in the first two lines. In effect, the main advantage of PCM is the small cell size. This allows a higher density and thus integrating more memory capacity in an embedded system. In addition, it offers a low static energy consumption, in contrast to DRAM that requires periodic refreshing. The read operation performance and energy are almost equivalent to those of DRAM. However, PCM presents one main disadvantage which is the costly write operation both in terms of energy and performance.

Several studies [2] investigated a hybrid memory to combine the advantages of DRAM and PCM. In this paper, we consider the horizontal integration of PCM, which is the most studied combination [2] in which the main memory is composed of a DRAM and a PCM at the samel level.

Memory type	DRAM	PCM
Cell size (F^2)	60-100	4-12
Static energy (mW/GB)	4	0
Read latency (ns)	10	20
Write latency (ns)	20	150
Write energy consumed per bit pJ/b)	2-3	14 - 20

Table 1: DRAM and PCM technology properties

2.2 Related Work

Existing DVFS strategies can be classified in: (1) application level techniques and (2) system level techniques.

Application level techniques: Here, the frequency scaling directives are achieved by the application. These directives are added off-line during the compilation phase [7] or on-line according to the tasks behavior [14].

System level techniques: At this level, applications are not aware of the frequency scaling. The system tunes the frequency according to several criteria. In the Linux kernel, for example, the higher the CPU utilization, the more the frequency is increased. In [3] and [6], the strategies rely on the memory-boundedness of tasks that can be defined as the ratio between memory access time and overall execution time. The higher the memory-boundedness, the lower the frequency is set. This is done in order to save energy without decreasing much the performance since memory access time is prominent and does not scale with frequency.

In our study, we are interested in system level techniques. To estimate the efficiency of our strategy, we used Workload Decomposition (WD) [3] as a reference in the evaluation part. The authors proposed to decompose the task execution time into an on-chip CPU time (T_{on}) and an off-chip memory access time (T_{off}). They used the ratio between both to scale the frequency according to the following equation:

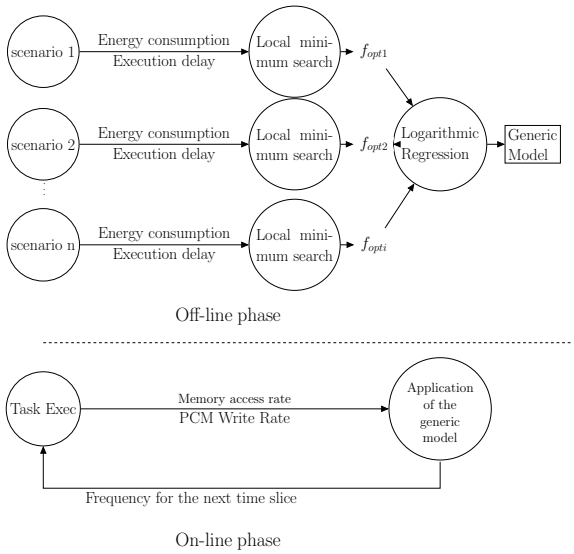


Figure 1: Overview of HyMAD strategy

$$f_{target} = \frac{f_{max}}{1 + PF_{loss}(1 + \frac{T_{off}}{T_{on}} \cdot \frac{f_{max}}{f^{t-1}})} \quad (1)$$

In eq. 1, the higher the memory boundedness ($\frac{T_{off}}{T_{on}}$), the more the frequency is reduced. PF_{loss} is used to tune the aggressiveness of the frequency scaling (from 0.05 to 0.2).

3. HYMAD DESIGN

3.1 Overview of the solution

As discussed in the related work Section, several DVFS strategies for memory-bound tasks have been designed. To the best of our knowledge, none has considered a hybrid memory. Given a data placement in a hybrid memory for a time period, HyMAD tunes the frequency of the CPU according to three parameters to trade off between energy and performance. Note that data placement into hybrid memory is out of the scope of this paper.

The basic idea behind HyMAD is to build an off-line frequency scaling model that minimizes the ED^2P, then to apply this model on-line to tune the frequency according to the task memory access pattern.

Figure 1 shows an overview of HyMAD off-line and on-line phases. The off-line phase is used to build the model. During this phase, we compute the execution time and energy consumption of running tasks with different memory scenarios. This is done relying on the embedded platform properties in terms of CPU, DRAM and PCM energy and performance properties. A memory scenario consists in defining the memory boundedness of a task in addition to the proportion of write operations operated on the PCM.

From the computed energy and execution time of different scenarios, and using a local minimum search, we determined the optimal frequency (f_{opt}) reduction that minimizes ED^2P.

Once this model is defined, we use it on-line to scale the frequency according to system needs. At the end of each time slice (t), we measure the memory access rate and PCM

write rate for the scheduled task. The generic model is applied with these measures to get the frequency to apply for the next time slice of the task.

3.2 System Model

In order to estimate the energy consumed by a hybrid memory system, we extended the model described in [3]. The overall execution time T_{exec} is composed of two parts, the on-chip CPU processing time T_{on}, and the off-chip time spent in main memory operations T_{off}. For simplicity, the CPU is assumed to be stalled during memory access.

$$T_{exec}(f) = T_{on}(f) + T_{off} \tag{2}$$

T_{exec} and T_{on} are frequency (f) dependent since the time spent on-chip depends on the number of execution cycles and the frequency. For a given number of cycles N and a number of cycles per instruction CPI, $T_{on}(f)$ is given by:

$$T_{on}(f) = \frac{N \cdot CPI}{f} \tag{3}$$

In our model, we distinguish between PCM and DRAM access times (T_{PCM} and T_{DRAM} respectively). Therefore, we subdivided the off-chip time as follows:

$$T_{off} = T_{DRAM} + T_{PCM} \tag{4}$$

T_{DRAM} and T_{PCM} include the times for both read and write operations. We assume that these times are equivalent on DRAM ($T_{r/w}^{DRAM}$). So the DRAM latency is related to the overall number of accesses (operations) N_{acc}. For PCM, we distinguish between the number of read N_r and write N_w operations (respective latency is T_r^{PCM} and T_w^{PCM}). So:

$$T_{off} = N_{acc} \cdot T_{r/w}^{DRAM} + N_r \cdot T_r^{PCM} + N_w \cdot T_w^{PCM} \tag{5}$$

The energy consumed by the system is also decomposed into on-chip and off-chip energy. The on-chip energy depends on the CPU frequency as it will be detailed here after.

$$E(f) = E_{off} + E_{on}(f) \tag{6}$$

The off-chip energy is composed of DRAM and PCM dynamic and static energy. As for the execution time, we did not distinguish between read and write operations for DRAM energy, but we did for PCM:

$$E_{off} = E_{stat}^{DRAM} + N_{acc} \cdot E_{r/w}^{DRAM} + N_r \cdot E_r^{PCM} + N_w \cdot E_w^{PCM} \tag{7}$$

For CPU, the energy is divided into static and dynamic:

$$E_{on}(f) = E_{stat}^{CPU} + E_{dyn}^{CPU}(f) \tag{8}$$

As in [8], we consider that the static energy does not depend on the frequency. The on-chip energy is given by [1]:

$$E_{on}(f) = E_{stat}^{CPU} + C_{eff} \cdot V^2 \cdot f \cdot T_{on} \tag{9}$$

where C_{eff} is the circuit effective capacitance, V the voltage and f the frequency. As in [1], we considered that the voltage is proportional to the frequency ($V = a \cdot f$), so the previous equation becomes:

$$E_{on}(f) = E_{stat}^{CPU} + C_{eff} \cdot a^2 \cdot f^3 \cdot T_{on} \tag{10}$$

We assume that the memory access behavior of a task is almost the same between two time slices allocated by the scheduler [3]. Thus, we used the behavior of the previous time slice to tune the frequency for the current one.

3.3 The off-line phase

The aim of the off-line phase is to determine a model of frequency scaling that adapts to the energy properties of the system. In this section, we first describe a formal model that allows to identify the frequency that minimizes ED^2P for a given data placement and memory boundedness (note that the same work can be done for EDP). Then, we apply this model to a case of study to show how to infer a generic model knowing the properties of an embedded system.

3.3.1 Identifying the optimal frequency

For a given data placement and memory boundedness, the optimal frequency that minimizes ED^2P is:

$$ED^2P(f) = E(f) \cdot T^2(f) \tag{11}$$

where $E(f)$ is the energy consumed during $T(f)$ when using the frequency f.

When the frequency is reduced from the maximum (f_{max}) supported by the CPU to a target frequency (f_{target}), the consumed energy $E(f_{target})$ is reduced with a proportion P_1 compared to the energy $E(f_{max})$:

$$E(f_{target}) = (1 - P_1) \cdot E(f_{max}) \tag{12}$$

However, the execution time is increased with a proportion P_2 compared to the execution time $T(f_{max})$:

$$T(f_{target}) = (1 + P_2) \cdot T(f_{max}) \tag{13}$$

Consequently and according to eq. 11:

$$ED^2P(f_{target}) = (1 - P_1) \cdot (1 + P_2)^2 \cdot ED^2P(f_{max}) \tag{14}$$

The optimal frequency minimizes the quantity $(1-P_1)\cdot(1+P_2)^2$. We note this quantity G.

In order to take into account the energy features of the system when identifying the optimal frequency, we develop in what follows the expressions of P_1 and P_2.

With the assumption that the static energy does not depend on the frequency [8], the energy reduction P_1 when the frequency is reduced from f_{max} to f_{target} can be given by:

$$P_1 = \frac{E_{dyn}^{CPU}(f_{max}) - E_{dyn}^{CPU}(f_{target})}{E(f_{max})} \tag{15}$$

Using eq. 10 and after replacing T_{on} by its detailed expression in eq. 3, we obtain the following expression for P_1:

$$P_1 = \frac{C_{eff} \cdot \alpha^2 \cdot f_{max}^2 \cdot N \cdot CPI}{E(f_{max})} \cdot (1 - (\frac{f_{target}}{f_{max}})^2) \tag{16}$$

Eq. 16 shows that the reduction of energy depends on:

- The ratio of the energy consumed by the CPU and the overall energy consumption. This dependency appears in eq. 16 by the following term that we call C:

$$C = \frac{C_{eff} \cdot \alpha^2 \cdot f_{max}^2 \cdot N \cdot CPI}{E(f_{max})} \tag{17}$$

- The ratio between f_{max} and f_{target} that we call R:

$$R = \frac{f_{target}}{f_{max}} \tag{18}$$

Then eq. 16 is simplified to:

$$P_1 = C \cdot (1 - R^2) \tag{19}$$

The value of P_2 is obtained using eq. 2. It is given by:

$$P_2 = \frac{T(f_{target}) - T(f_{max})}{T(f_{max})} \quad (20)$$

Since the memory access time does not depend on the frequency, T_{off} is constant, eq. 20 is then simplified to:

$$P_2 = \frac{T_{on}(f_{target}) - T_{on}(f_{max})}{T(f_{max})} \quad (21)$$

After replacing T_{on} by the expression given in eq. 3, we obtain the following expression for P_2:

$$P_2 = \frac{N \cdot CPI}{f_{max} \cdot T(f_{max})} \cdot (\frac{f_{max}}{f_{target}} - 1) \quad (22)$$

From Eq. 22, the increase of execution time depends on:

- The ratio between the processing time and the overall execution time, shown by the following term L:

$$L = \frac{N \cdot CPI}{f_{max} \cdot T(f_{max})} \quad (23)$$

  Thus, the increase of execution time depends on the memory boundedness of the running tasks. The greater the processing time, the higher P_2.

- The ratio R between f_{max} and f_{target}.

Then:

$$P_2 = L \cdot (\frac{1}{R} - 1) \quad (24)$$

Equations 16 and 22 show that the quantity G to minimize is closely related to the energy features of the system and the memory boundedness of the running tasks. Once the expressions of P_1 and P_2 detailed, G can be expressed as:

$$G(R) = \frac{1}{R^2} \cdot (L^2 - CL^2) + \frac{1}{R} \cdot (2L - 2L^2 + 2CL^2 - 2CL) + 1 + L^2 - 2L - C + R \cdot (2CL - 2CL^2) + R^2 \cdot (C + CL^2 - 2CL) \quad (25)$$

Thus, we can identify the value of R that minimizes ED^2P for a given scenario. The optimal reduction R_{opt} is obtained when the first derivative of the G function according to R is zero. Since the frequency of the processor cannot be equal to zero, R never equals zero. This allows to simplify the equation that R_{opt} must satisfy to:

$$R^4 \cdot (2C + 2CL^2 - 4CL) + R^3 \cdot (2CL - 2CL^2) + R \cdot (2L^2 - 2L - 2CL^2 + 2CL) + 2CL^2 - 2L^2 = 0 \quad (26)$$

From the obtained optimal reduction R_{opt}, we can determine the optimal frequency f_{opt} to set for the next time slice.

Using eq. 26 for frequency scaling is cumbersome as it requires to determine C and L which requires complex energy measurements. Thus, we built a generic model that approaches the optimal frequency reduction of eq. 26. This explained in the next section.

3.3.2 Building the frequency scaling model

In order to define a generic close to optimal model for frequency scaling according to the system energy properties, we determine, for a set of scenarios, the optimal frequency using eq. 26. We can then apply regression to obtain a simpler generic model for frequency scaling.

β_0	β_1	β_2
-0.52	-0.38	-0.11

Table 2: Model instantiation

According to our assumption that both memory access rate and PCM write rate should be considered, and after testing the model with RaspBerry and BeagleBone Black boards, we observed that the frequency reduction follows a logarithmic shape, thus, we propose the following model:

$$R = e^{\frac{PS-100}{39} \cdot (\beta_0 + \beta_1 R_{mem} + \beta_2 R_{wpcm})} \quad (27)$$

Where R_{mem} represents the rate of memory accesses:

$$R_{mem} = \frac{T_{off}}{T} \quad (28)$$

R_{wpcm} represents the rate of PCM writes:

$$R_{wpcm} = \frac{N_w \cdot T_w^{PCM}}{T} \quad (29)$$

PS is the static priority of the task. Priority interval for conventional tasks is from 100 to 139 (lower value is the higher priority). In eq.27, when PS is 100, R is 1 which means that the frequency is set to the maximum value f_{max}. The higher PS, the more the frequency can be reduced.

3.3.3 Instantiating the frequency scaling model: the RaspBerry Model-A case study

In order to evaluate the efficiency of the regression model that we proposed, we applied it to the RaspBerry Model-A system. Knowing its energy features from the data sheet [9], we built several scenarios. Each scenario consists of the memory boundedness of task, the percentage of write operations performed on PCM, the energy consumed by the CPU and the hybrid memory and the processing and memory times. We then determined for each scenario the optimal frequency value using eq. 26.

From those calculated frequencies, we performed a regression according to the model given in eq. 27 to get the values of β_0, β_1 and β_2. The results are given in Table 2.

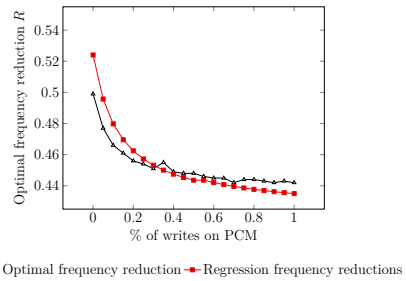


Figure 2: Theoretical and regression model comparison

We compared the frequency reductions given by the obtained regression model and the optimal frequency reductions obtained with the theoretical model proposed in Section 3.3.1. Results are shown in Figure 2. We note that frequency reductions given by the regression model are very close to the optimal frequency reductions from eq. 26.

3.4 The on-line phase

Once the model is defined, it is used to scale the frequency according to the system needs. the memory access rate R_{mem} and the PCM write rate R_{wpcm} are measured. Knowing these two parameters and the static priority of the task to run in the next time slice, the frequency for current slice is scaled using the model of eq. 27.

4. EVALUATION

This section presents the evaluation of HyMAD. First, we describe the methodology used, then, we discuss the results.

4.1 Experimentation Setup

For the following experimentations, we used the energy properties of the RaspBerry Model A. The evaluations were performed using synthetic applications, where the memory access rate and the percentage of writes on PCM were varied.

4.2 Performance Evaluation Methodology

We mainly used two metrics for comparison, the Energy-Delay and Energy-Delay^2 products (EDP and ED^2P).

Our experiments were performed in three steps:

(1) We evaluated the relevance of considering both memories (PCM and DRAM) explicitly in the model. To do so, we compared to three strategies :

1. A model that was built the same as HyMAD, but without specifically considering PCM. We assumed that all memory accesses consume the same energy amount. Consequently, only R_{mem} was considered. The energy consumption of an operation was fixed to the one of a DRAM operation. The regression gives the following frequency scaling model:

$$R = e^{\frac{PS-100}{39}\cdot(-0.51-0.71\cdot R_{mem})} \quad (30)$$

   In what follows, this model is called MAD (**M**emory **A**ccess **D**VFS strategy).

2. Workload decomposition model (WD) [3], see the related work section.

3. A DVFS-free system where the frequency is always set to the maximum value.

HyMAD, MAD and WD were applied on three applications with memory boundedness of 0.1, 0,5 and 0.9. The experiment is performed with a PCM write energy of 17.5pJ [5]. In order to better show the impact of the PCM, we also tested with the energy consumption of PCM writes to 70pJ .

(2) To test if the HyMAD model works fine even when PCM is not accessed, we considered workloads that do not perform memory operations on the PCM and compared HyMAD to WD which was designed for a DRAM-only systems.

(3) The last experiment was achieved to observe the desired impact of the static priority (PS) on HyMAD.

4.3 Results and Discussion

1) Impact of considering the PCM: Figures 3a, 3b and 3c show by how much HyMAD enhances MAD for the ED^{2}P metric. We notice that HyMAD does not reduce the frequency as much as MAD. We also observe that HyMAD enhances ED^{2}P metric by 2-8% as compared to MAD. This means that the optimal frequency reduction in the hybrid memory case is different than in the DRAM-only case. The effectiveness of HyMAD as compared to MAD is due to the fact that MAD considers that all memory accesses consume the same amount of energy. Thus, it is important to consider the heterogeneity of the memory to better scale the frequency. When comparing HyMAD to WD, we observe that HyMAD enhances ED^{2}P up to 20% according to the PCM write rate. When compared to a DVFS-free system, HyMAD enhances the ED^{2}P value between 2-45%.

Figure 4 shows the results of the same experiment while varying the PCM write energy to 70pJ. The enhancement performed by HyMAD as compared to MAD is 2-14 % and 3-15% as compared to WD.

2) HyMAD on PCM-free workloads: Figure 5 shows the comparison of HyMAD with WD in case of a workloads not accessing the PCM. We notice that HyMAD gives better performance for both EDP and ED^{2}P metrics. It enhances EDP as compared to WD by 5-33 %. ED^{2}P enhancement is 1-15 %. This means that our methodology is effective even for an NVM-free embedded platform.

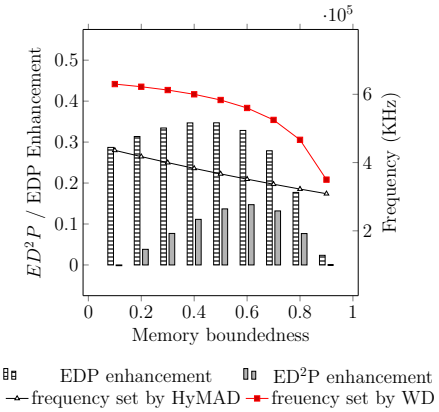


Figure 5: HyMAD EDP and $ED2P$ as compared to WD for PCM-free workloads

3) Impact of static priority: Figure 6 shows the frequency scaling according to the static priority. We experimented with three values of PS: 100 (high priority task), 120 (medium priority task) and 139 (a low priority task). HyMAD does not reduce the frequency for a high priority task. The lower the priority, the more the frequency can be scaled.

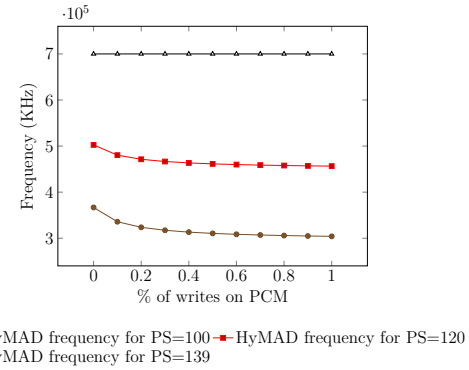
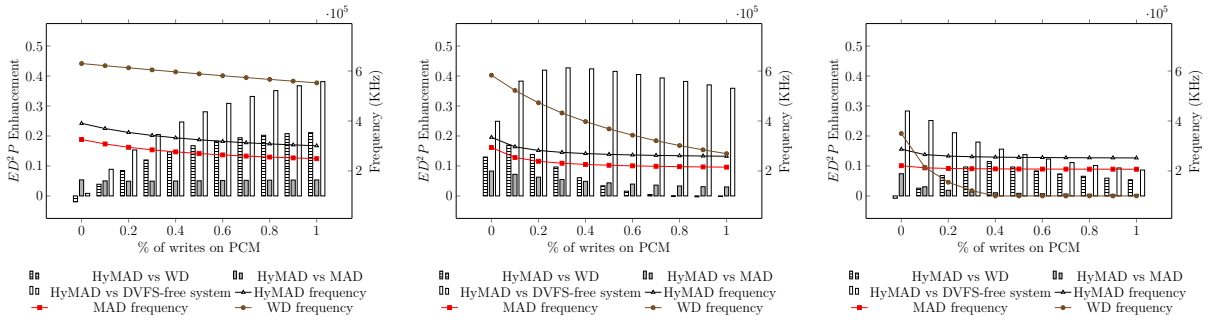
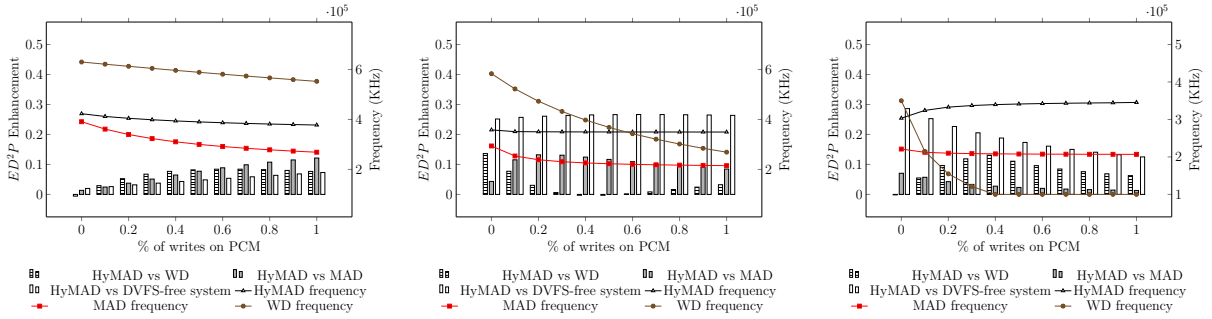


Figure 6: Impact of static priority on HyMAD



(a) R_{mem}=0.1 (b) R_{mem}=0.5 (c) R_{mem}=0.9

Figure 3: HyMAD $ED2P$ enhancements over MAD and WD with different R_{mem} and a PCM write consumption of 17.5pJ



(a) R_{mem}=0.1 (b) R_{mem}=0.5 (c) R_{mem}=0.9

Figure 4: HyMAD $ED2P$ enhancements over MAD and WD with different R_{mem} and a PCM write consumption of 70pJ

5. CONCLUSION

This paper presents HyMAD, a hybrid memory-aware DVFS strategy. The main idea behind HyMAD is to consider the heterogeneity of a memory in order to design an efficient DVFS policy. In effect, as PCM energy consumption is very high in case of write operations, the optimal behavior to adopt is different from the case of a DRAM-only architecture. HyMAD aims to determine the optimal behavior of frequency scaling adapted for a given hardware platform.For future work, we would like to investigate ways to couple HyMAD with a data placement strategy at the operating system level for energy/performance trade-off optimization.

6. REFERENCES

[1] Y. Benmoussa, E. Senn, N. Derouineau, N. Tizon, and J. Boukhobza. Joint dvfs and parallelism for energy efficient and low latency software video decoding. *IEEE TPDS*, 29(4):858–872, April 2018.

[2] J. Boukhobza, S. Rubini, R. Chen, and Z. Shao. Emerging NVM: A survey on architectural integration and research challenges. *ACM TODAES*, 23(2):14:1–14:32, 2017.

[3] K. Choi, R. Soma, and M. Pedram. Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times. *IEEE TCAD*, 24(1):18–28, 2005.

[4] R. Cochran, C. Hankendi, A. Coskun, and S. Reda. Identifying the optimal energy-efficient operating points of parallel workloads. In *ICCAD*, pages 608–615, Nov 2011.

[5] G. Dhiman, R. Ayoub, and T. Rosing. PDRAM: A hybrid PRAM and DRAM main memory system. In *ACM/IEEE DAC*, pages 664–669, 2009.

[6] C.-H. Hsu and W.-C. Feng. A power-aware run-time system for high-performance computing. *SC*, 2005.

[7] C.-H. Hsu, U. Kremer, and M. S. Hsiao. Compiler-directed dynamic frequency and voltage scheduling. In *PACS*, 2000.

[8] N. S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan. Leakage current: Moore's law meets static power. *Computer*, 36(12):68–75, Dec 2003.

[9] R. P. T. Ltd. Raspberry pi2 – power and performance measurement, 2016, (Accessed September 20, 2018).

[10] S. Mittal. Energy saving techniques for phase change memory (PCM). *arXiv preprint arXiv:1309.3785*, 2013.

[11] O. Mutlu. Memory scaling: A systems architecture perspective, Aug. 2013.

[12] P. Ranganathan. From microprocessors to nanostores: Rethinking data-centric systems. *Computer*, 44(1):39–48, Jan 2011.

[13] J. S. Vetter and S. Mittal. Opportunities for nonvolatile memory systems in extreme-scale high performance computing. *CiSE*, 17(2):73–82, 2015.

[14] Q. Wu, M. Martonosi, D. W. Clark, V. J. Reddi, D. Connors, Y. Wu, J. Lee, and D. Brooks. A dynamic compilation framework for controlling microprocessor energy and performance. In *IEEE/ACM MICRO*, pages 271–282, 2005.