# DVFS Space Exploration in Power Constrained Processing-in-Memory Systems

Marko Scrbak[1], Joseph L. Greathouse[2], Nuwan Jayasena[2], and Krishna Kavi[1]

[1] University of North Texas, Denton, TX, USA
markoscrbak@my.unt.edu, krishna.kavi@unt.edu
[2] Advanced Micro Devices, Inc. (AMD), Sunnyvale, California, USA
joseph.greathouse@amd.com, nuwan.jayasena@amd.com

**Abstract.** In order to deliver high performance under stringent power constraints, future systems may include die-stacked memories with processing-in-memory (PIM) cores. Because of their proximity to the memory, PIMs are expected to target applications which require high bandwidth, implying that PIMs do not need the same computational capabilities as traditional host processor and can therefore be implemented using slower, low-leakage transistors to increase energy efficiency. Such systems must carefully balance design-time choices, such as the circuits used to build the devices, and run-time choices, such as DVFS states and the preferred hardware platform on which to run the application. This paper explores these parameters in a GPGPU PIM system with a large compute-optimized host and a collection of bandwidth-optimized PIMs. We develop high-level performance and power models and use them to find optimal DVFS and kernel placement decisions for a series of GPGPU applications targeting maximum energy efficiency. We find, for instance, that the energy efficiency of PIM systems is greatly affected by DVFS; simply selecting the optimum hardware (host/PIM) results in 7× higher $ED^2$ than migrating work in conjunction with DVFS.

**Keywords:** Processing-in-Memory, DVFS, GPGPU, High Performance Computing, Energy Efficiency, Computer Architecture, 3D-DRAM

## 1 Introduction

With the breakdown of Dennard scaling, architects rely on increasingly sophisticated dynamic voltage and frequency scaling (DVFS) controls to optimize performance while meeting stringent power, energy, and thermal targets [2, 5, 15, 19]. For related reasons, modern processors are increasingly adding heterogeneous accelerators, most commonly GPUs [10, 20]. As techniques such as DVFS and heterogeneous processing increase the computational efficiency of processors, the memory system is becoming a growing performance and energy bottleneck [4]. Recent stacked memory technologies, such as Hybrid Memory Cube (HMC) [16] and High Bandwidth Memory (HBM) [9], can provide higher bandwidth and reduced access energy compared to traditional technologies such as DDR4. Some

researchers propose to use 3D die stacking to place cores in a logic layer under the memory dies and migrate some computation closer to memory, a configuration referred to as processing-in-memory (PIM) [1, 7, 13, 17, 21, 23]. In these studies, a collection of PIM devices are used along with a traditional "host" processor [17, 18, 21, 23]. The host can be optimized for traditional compute-heavy workloads and can be implemented using high-performance transistors. In contrast, PIMs can be optimized for bandwidth-intensive workloads, can be implemented using lower-power circuits, and can be smaller and run at lower frequencies. This can significantly increase energy efficiency and reduce difficulties from tight PIM thermal constraints [6].

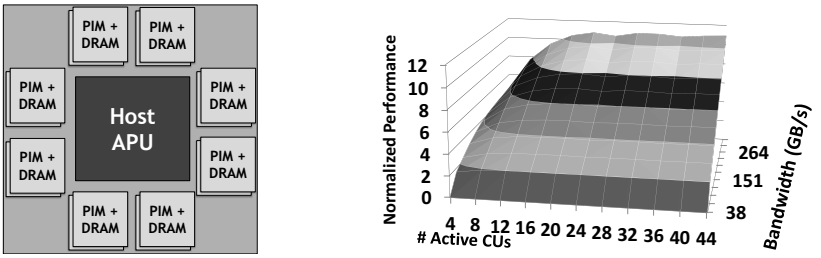
In this paper, we evaluate the effects of DVFS on the power and performance of a PIM-enabled system while taking into account the architectural and process differences between a compute-focused host and PIMs optimized for memory bandwidth. We develop power and performance models which capture the key differences between these devices and use them to study the design space. Our findings show that running some compute-intensive kernels on low-frequency host cores decreases $ED^2$ by 2×, while others see up to 4.5× benefit from the lower-powered PIMs. Memory-intensive applications can reduce $ED^2$ 10× compared to the host when the PIMs can run at lower frequencies. Compared to simply migrating workloads, making DVFS decisions in conjunction with migration decisions can result in a 7× reduction in $ED^2$. In addition, a mix of DVFS and workload migration can allow applications to achieve higher performance under tight power caps. To the best of our knowledge, we are the first to explore the impact of DVFS on power and performance of GPU-based PIM systems. The key contributions of this paper are:

- An analytical power model which captures the differences in leakage and dynamic power of host and PIM devices.
- An evaluation of optimal hardware platform (host vs. PIM) for OpenCL™ kernels in order to achieve maximum energy efficiency.
- Evaluation of maximum achievable performance under different power constraints.

The rest of the paper is organized as follows. In Section 2, we explain the system organization. Section 3 describes the performance and power models. Section 4 describes the details of our methodology and experimental setup. In Section 5, we present our results and discuss the findings. Section 6 covers the related work and Section 7 concludes the paper.

## 2 Baseline System organization

Figure 1(a) shows an illustration of the system we study. At the heart of the node is a heterogeneous chip containing both a CPU and GPU. This design is also equipped with 3D-stacked DRAM [9, 16], which can deliver the high bandwidth necessary for the on-chip GPU. 3D die stacking allows for tight integration of optimized DRAM and logic dies. As such, it allows near-data computing in the




(a) A heterogeneous processor consisting of a host APU and multiple 3D DRAM stacks with PIMs. PIMs are low-power GPGPU cores and as such can utilize the high in-stack memory bandwidth

(b) *DeviceMemory.readGlobalMemoryCoalesced* scaling curve. Performance scales with bandwidth and stagnates with increase in compute power. Such a kernel is classified as memory bound.

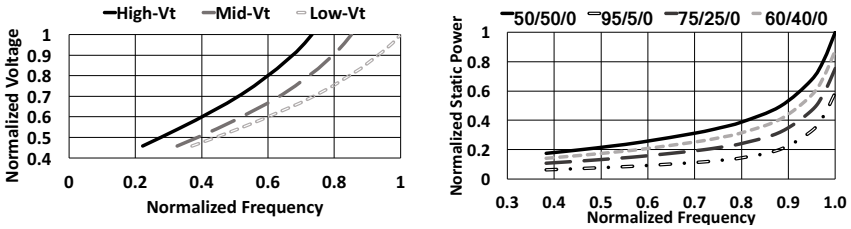
**Fig. 1.** Baseline system (a) and kernel performance scaling on GPUs (b)

form of PIM, where computation devices embedded on the logic layer can utilize the high in-stack bandwidth to improve the performance of memory-intensive applications [1, 17, 21, 23]. A multitude of PIM designs have been explored, such as PIMs built from low-power CPU cores [17, 21], GPGPUs [23], and reconfigurable logic [7]. While low-power CPU PIMs were shown to be very energy efficient, they were able to utilize only a fraction of the available in-stack memory bandwidth [8]. A multitude of CPU cores could push more bandwidth but would exceed the power and thermal envelope of the memory stack. Zhang et al. proposed a PIM architecture based on GPU accelerators (GPGPUs) [23]. These highly multi-threaded vector processors are suitable for running highly-parallel application kernels and provide high computational and memory throughput. At the same time, the GPU compute units are simple in-order cores, which makes the architecture energy efficient and provides opportunity for embedding them in the logic layer of 3D-stacked DRAM. Furthermore, mature programming models (e.g., OpenCL$^{TM}$) ease the programmability of such devices. In this study, we focus on GPUs as PIMs and compare their power and performance to the host on-chip GPUs. In a heterogeneous PIM system, GPU PIM compute units will be aimed at improving the energy efficiency and performance of memory-intensive code. The host GPU will primarily be used for compute-intensive code, since it has less memory bandwidth and more compute resources than the PIMs.

## 3 Performance and Power Models

### 3.1 Performance Model

In order to assess the performance for our target hardware configurations, we rely on an analytical GPU performance modeling framework [22]. Such models are more effective for large design space explorations than cycle-level simulators, and they capture enough detail to reasonably estimate future hardware performance [14].




(a) V/f curves for three different types of devices. Axes are normalized to their maximum values

(b) Relative leakage values for different Vt distributions.

**Fig. 2.** V/f characteristics and relative leakage power for different types of devices.

The model described in [22] has between 10% and 30% error, which is comparable to state-of-the-art system simulators [3]. The idea is to run a series of GPU kernels on native hardware and collect performance statistics for each kernel invocation for different hardware configurations. This allows us to obtain knowledge about how each kernel's performance scales with hardware resources such as frequency, memory bandwidth and compute units. An example of kernel scaling characteristic is shown in Figure 1(b).

Our model was built from over 200 kernels at 720 different hardware configurations across various frequencies, memory bandwidths and number of active compute units on an AMD FirePro$^{TM}$ W9100 GPU. The gathered scaling statistics are clustered into groups with similar scaling characteristics. A machine learning model is then trained which is later used to classify previously unseen kernels into one of the scaling groups. We finally use the kernel scaling characteristic to extrapolate kernel performance from a native hardware configuration to a desired target hardware configuration (host/PIM).

### 3.2 Power Model

The total power for host/PIM devices is calculated as the sum of chip dynamic power and leakage power. We modeled dynamic and leakage power in a way that they capture host and PIM architectural differences (number of CUs, frequency) and differences in process technology (host - high performance process, PIM - low power process). The difference in process technology affects the voltage/frequency (V/f) characteristics of the devices which, in turn, affects dynamic and leakage power. Our models assume a feature size of 14nm.

**DVFS Characteristics** Modern computer chips are designed using multiple types of transistors, i.e. a mixture of low-, medium-, and high-threshold transistors, to target different design tradeoffs, e.g. high-performance vs. low power.

Low-threshold voltage (Low-Vt) devices are used in timing-critical paths, but have high leakage power. High-threshold voltage (High-Vt) devices have

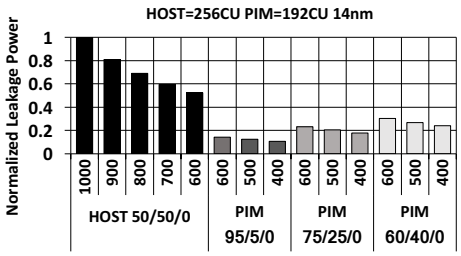



**Fig. 3.** Leakage power values for host and PIM with different Vt distributions. A low-power PIM with minimal leakage power (such as PIM-95/5) would be desirable in order to minimize total PIM power consumption.

low leakage power but are slower, and are typically used in circuits that are off the timing-critical paths. Medium-threshold voltage (Mid-Vt) devices offer a tradeoff between High-Vt and Low-Vt devices by having medium power requirements and medium delay. In general, low power chips are designed using a larger percentage of High-Vt devices and high-performance chips with a larger percentage of Mid-Vt and Low-Vt devices. The host processor is assumed to execute compute-intensive code and will therefore be a high-performance device. PIMs are assumed to be an equivalent of a low-power device, and will have significantly lower leakage power than the host and run on lower frequencies and consume less dynamic power.

Figure 2(a) shows V/f characteristics of three different types of devices for a 14nm process. Instead of modelling a V/f characteristic of a design with a specific Vt distribution, we chose the V/f characteristics of both the host and the PIMs to be equivalent to that of Mid-Vt devices and limit the operating frequency ranges for host and PIM. We used this method because the Vt ratios change based on the process maturity and process variation and thus it would be impractical to model specific V/f curves for host/PIM. We choose the operating frequency range for host to be 600MHz-1000Mhz and that of PIM 400MHz-600MHz. We also study the impact on leakage power if the frequency ranges require a different mix than the nominal one we picked.

**Leakage Power** Leakage power depends on operating voltage, temperature, and the ratios of devices used in a chip design. We pick the ratios for host to be representative of a high performance GPU with a 50/50 High-Vt/Mid-Vt distribution. We performed a parameter sweep of three different device ratios for the PIMs and observed their impact on leakage power. High-Vt/Mid-Vt ratios included in the model are 60/40, 75/25, and 95/5 respectively. We model the leakage power by estimating the leakage power value of host at the highest voltage-frequency (V/f) point and then scale it to other V/f points using relative leakage values between host and PIM. The relative leakage values between different ratios are obtained from a circuit design tool. Figure 2(b) shows the relative

leakage values for four different Vt distributions studied. Derived leakage power numbers are shown in Figure 3. At identical frequencies (600MHz) PIM devices will have up to 3× lower leakage than the host.

It is important to target low-leakage PIM designs for several reasons. First, the power dissipation needs to be minimal so as to not exceed the 3d memory stacks' power and thermal limitations. Second, a higher power dissipation would increase the stack temperature and the DRAM refresh rate. Third, most of the applications executed on PIM will have lower dynamic power and therefore leakage power will take up a significant portion in PIM total power. For the rest of the paper we assume a 95/5 High-Vt/Mid-Vt distribution for the PIMs.

**Dynamic Power** The dynamic power of the host and PIM devices is a function of the target hardware configuration (frequency, voltage, number of CUs) as well as the running kernels switching activity. We calculate the dynamic power of host/PIM devices by scaling the dynamic power of a base hardware configuration such as an AMD FirePro™ W9100 GPU to a desired target hardware configuration (host/PIM - Table 1) using the following equation:

$$P_{dynamic} = MAXDP * \frac{CU_{target}}{CU_{base}} * \frac{f_{target}}{f_{base}} * \frac{V_{target}^2}{V_{base}^2} * C_{AC} * C_{scaled} \quad (1)$$

The idea is to scale a known *maximum dynamic power (MAXDP)* consumed by a high-end GPU (such as the AMD FirePro W9100 GPU) at a given frequency, voltage, and number of compute units, to a target hardware configuration *(f, V, CUs)*. The assumption is that the PIM and host CUs will be architecturally similar to present high-end GPUs, and therefore the maximum dynamic power consumed per each CU will be roughly the same for the same feature size, frequency and voltage. This way we can estimate the maximum dynamic power consumed by the target hardware, i.e. at 100% switching activity. The actual dynamic power consumed by a GPU kernel will depend on the chip switching activity (shown as $C_{AC}$ in equation 1) during kernel execution. The total dynamic power will therefore be a fraction of the target maximum dynamic power. Target capacitance (as compared to the base GPU capacitance) will also be lower and ultimately will lower the target dynamic power. We factor this in as the last element of the equation $C_{scaled}$.

## 4 Methodology and Experimental Setup

### 4.1 Target Hardware Baseline

We assume the target node, as depicted in Figure 1, will have a high-performance host APU and eight 3D DRAM stacks with low-power GPU PIM cores. Details of the target system are listed in Table 1. We set the PIMs' aggregate memory bandwidth to be 2× higher than the host's, assuming that only 50% of the possible in-stack bandwidth will be available to the host due to the high power

**Table 1.** Target system parameters

| | host | PIM |
|---|---|---|
| # of CUs | 256 | 192 (8x24) |
| Mem. Bandwidth | 1 TB/s | 2 TB/s |
| Frequency (MHz) | 600-1000 | 400-600 |
| Tech. Node | 14nm FinFET | 14nm FinFET |
| Process | high-performance | low-power |

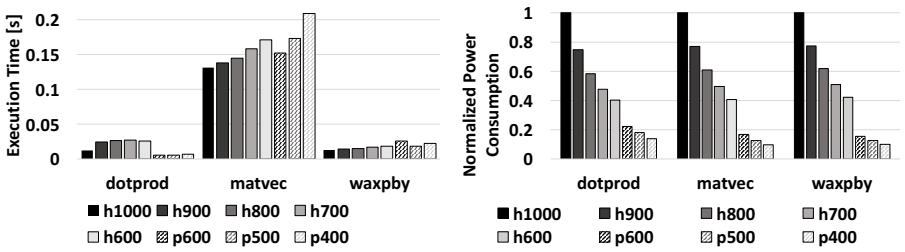
consumption of the active links needed to support high off-chip memory bandwidth. Each of the eight PIM stacks is assumed to have 24 embedded low-power GPU CUs, for a total of 192 PIMs. The host APU is assumed to have 256 GPU Compute Units (CUs) and has more compute power than the eight PIM stacks. Such a high number of CUs is an optimistic assumption, however it will be achievable with future technology scaling.

### 4.2 Benchmark Selection

We selected 15 applications from a wide range of publicly available GPU benchmark suites. These benchmarks tend to exhibit enough parallelism to utilize, and are expected to scale to, the target hardware configurations. In addition, the benchmarks selected rely on algorithms for which we can easily split data and tasks such that the PIMs primarily access their local DRAM stacks. The 15 selected applications are categorized based on their performance scaling characteristics [12]. Compute bound benchmarks - *lavaMD, NBody, MonteCarloAsian, MaxFlops, CoMD* - contain mostly kernels which scale with compute resources. Memory bound benchmarks - *kmeans, MatrixTranspose, miniFE, DeviceMemory* - contain mostly kernels which scale with memory bandwidth. Balanced benchmarks - *b+tree, MatrixMultiplication* exhibit different scaling behavior for different compute/bandwidth ratios. We also select benchmarks which have a mix of compute/memory/balanced kernels - *backprop, GEMM, BoxFilter, XSbench* to show the impact of kernel placement on total runtime/power consumption when kernels have different scaling characteristics.

### 4.3 Experiments

We collected performance counters for each kernel invocation of the 15 benchmarks and estimate their power/performance using previously described power and performance models. We also include the energy spent on memory accesses. We evaluated the optimal hardware choices to run the kernels when using DVFS and compared them to cases where all benchmark kernels run on either the host or PIM devices, while targeting maximum energy efficiency (minimum $ED^2$). Figure 4 shows the motivation of the potential tradeoffs. We also evaluated the maximum performance under power constraints and show how optimal hardware choice shifts to PIMs, which consume significantly less power. It is assumed that




(a) miniFE kernel total execution times (b) miniFE kernels power consumption

**Fig. 4.** miniFE kernels runtime (a) and power consumption (b) for host 600MHz-1000MHz and PIM 400MHz-600MHz

once a kernel is running on the host/PIM it will remain there (i.e., it will not migrate between devices) for all invocations of that kernel.

## 5 Results and Analysis

For the purpose of analyzing the energy efficiency we evaluated the energy-delay$^2$ metric because it represents a tradeoff between energy and performance. Figure 5 compares the ED$^2$ value for ED$^2$ optimal placement with ED$^2$ of host-only and PIM-only placement. We observe that for highly compute intensive benchmarks like *MaxFlops* and *NBody*, the host has significantly better ED$^2$ and is the optimal choice. Interestingly, these two applications achieve minimum ED$^2$ when running at the highest DVFS points. This implies that PIMs aren't necessarily the most energy-efficient choice for computation. Other benchmarks have better ED$^2$ when all kernels run on the PIMs, except *CoMD* where 2 kernels have better ED$^2$ when running on the PIMs while the others are best on the host. Figure 5 shows how PIMs are an optimal choice in many cases when targeting energy efficiency. This shows that the addition of PIMs to a heterogeneous node

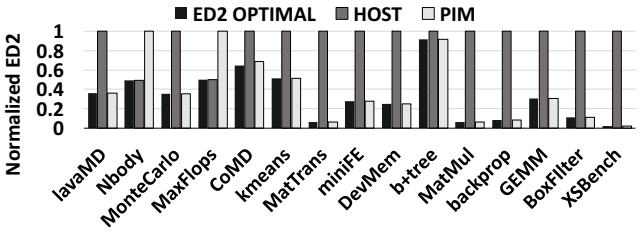
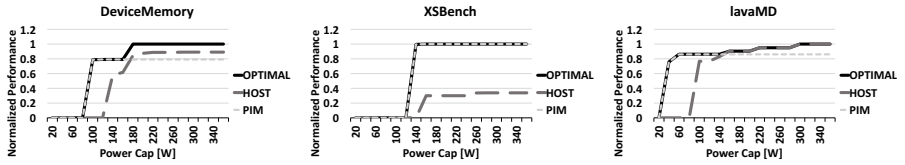



**Fig. 5.** ED$^2$ comparison when trying to optimize for minimum ED$^2$. Very high compute intensive applications achieve minimum ED$^2$ while running on host. This means that host will be more energy efficient for such applications than PIM regardless of the higher power consumption.




(a) DeviceMemory (b) XSBench (c) lavaMD

**Fig. 6.** Maximum Performance under Power Constraints

architecture can yield high throughput and high energy efficiency even when compared to host running at lower DVFS states. In many cases, including applications that are somewhat compute-bound, the work would move to PIM, which significantly reduces power at the expense of small performance loss. Exceptions are highly compute intensive applications like *b+tree, MaxFlops, and NBody.*

### 5.1 Maximum Performance under Power Constraints

The host can deliver higher performance for applications that are very compute intensive. The question remains whether this will remain true under power constraints, and at which point the PIMs will deliver better performance than the host. To confirm that PIMs are in many cases indeed a better choice, even when host is running at a lower DVFS state, we evaluated what is the maximum performance we can get from a benchmark when each kernel consumes less power than a specified power constraint. There will be a performance optimal hardware choice for each kernel, and this will change depending on the power limit. Figure 6 compares the maximum performance under different power caps for 3 benchmarks. We show a subset of all benchmarks because others show similar behavior and same conclusions can be made. We see that the host always consumes at least 100W (at the lowest DVFS state) and cannot perform under power constraints lower than 100W. PIMs can on the other hand deliver

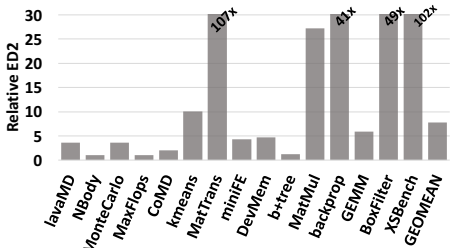



**Fig. 7.** Comparison of $ED^2$ values when host and PIM employ DVFS with a case where host and PIM run only at highest DVFS states (host-1000MHz, PIM-600MHz). By using DVFS in conjunction with PIMs we can on average improve the minimum achievable $ED^2$ by 7x, and in cases of memory intensive benchmarks by 40x-100x.

good performance even under tight power budgets due to their low power consumption. In cases of memory bound benchmarks like *XSBench*, PIMs always deliver significantly higher performance due to higher memory bandwidth. An interesting case is *lavaMD*, where the PIM outperforms the host at some intermediate DVFS state. This is because the application can compensate the lower performance of lower DVFS state by exploiting higher memory bandwidth.

### 5.2 Discussion

When optimizing applications for maximum energy efficiency, benchmarks which consist of heavily compute intensive kernels achieve 2× lower $ED^2$ when running on host at highest DVFS states (1000MHz). However, other compute intensive kernels achieve lower $ED^2$ when running on PIM (1.5× - 4.5× lower than on host), while suffering minimum performance losses (20%-50%) over performance optimal case. While optimal hardware choices play a significant role, the addition of DVFS to the system proves to be crucial in maximizing the energy efficiency. Figure 7 compares the minimum achievable $ED^2$ value of a host/PIM system with DVFS to a host/PIM system without DVFS (running on highest DVFS state). We can see that by using DVFS to complement the already energy-efficient system design we can on average improve the energy efficiency by 7×, and in some cases between 40×-100×. When optimizing applications for maximum performance for systems with lower power budgets we can achieve 1.2× - 2.5× better performance if we pick the right hardware (host/ PIM) and allow for DVFS. For small power budgets, PIMs can achieve better performance than the host at a lower DVFS state, while for power budgets lower than 100W, the host would exceed the power limit while PIMs would be able to remain operational and deliver performance comparable to host for a fraction of power consumed. Our findings strengthen the hypothesis of PIMs being a useful heterogeneous platform and show the importance of DVFS as a mean to maximize performance and energy efficiency in HPC systems with PIM. Additionally, our study allows for future exploration of optimizations when multiple applications are executing in the system by trading off power and performance of different applications to achieve combined optimum performance gains.

## 6 Related Work

Zhang et al. [23] proposed a PIM architecture based on GPGPUs and evaluated the performance and power benefits of such systems. However, this work only considered a single operating point for the host and PIMs, and evaluated host and PIM execution in isolation. We extend this work to include the characterization of the impact of DVFS and co-optimization of both host and PIM. In addition, we created power models which can capture the differences between host and PIM and evaluate the system on application level and not just kernel level.

Schulte et al. [11] investigated the effect of varying engine frequency/voltage, memory bandwidth and number of compute units on GPU performance and power. The authors explored DVFS for standalone GPUs. We instead consider

the presence and co-optimization of heterogeneous execution engines (i.e., high-performance host and low-power PIM).

Ščrbak et al. [21] explored a variety of design choices in ARM-based PIM systems, including caches, frequency/voltage and their effect on the overall energy efficiency of the system. The research remained focused on ARM-based PIM architectures and doesn't explore GPUs as an alternative.

## 7 Conclusion

In this paper we explored the effects of DVFS on energy efficiency of a GPU PIM system while accounting for architectural and process differences of the host and PIM devices. We developed analytical power and performance models to capture these differences and use them to explore the PIM DVFS design space. Our findings show that a PIM system with DVFS is more energy-efficient than a PIM system without DVFS, and results in 7× lower $ED^2$ values on average. By utilizing DVFS for host we can additionally decrease $ED^2$ by 2× for compute-intensive applications and by 4.5× when using DVFS with PIMs. Furthermore, when using DVFS and low-power PIMs and optimizing performance for systems with tight power budgets, we can achieve 1.2×-2.5× better performance if we pick the right hardware (host/PIM) and DVFS point. Our study allows for future exploration of optimizations when multiple applications are simultaneously executing in the system. We will evaluate such optimizations in future work.



## References

1. Ahn, J., Hong, S., Yoo, S., Mutlu, O., Choi, K.: A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing. In: Proc. of the Int'l Symp. on Computer Architecture (ISCA) (2015)
2. Akram, S., Sartor, J.B., Eeckhout, L.: DVFS Performance Prediction for Managed Multithreaded Applications. In: Int'l Symp. on Performance Analysis of Systems and Software (ISPASS) (2016)
3. Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M.D., Wood, D.A.: The gem5 simulator. ACM SIGARCH Computer Architecture News 39(2), 1–7 (2011)
4. Black, B.: Die Stacking is Happening! Presented at MICRO (2013)
5. Cochran, R., Hankendi, C., Coskun, A.K., Reda, S.: Pack &Cap: Adaptive DVFS and Thread Packing under Power Caps. In: Proc. of the Int'l Symp. on Microarchitecture (MICRO) (2011)
6. Eckert, Y., Jayasena, N., Loh, G.H.: Thermal Feasibility of Die-Stacked Processing in Memory. In: Workshop on Near-Data Processing (WoNDP) (2014)

7. Farmahini-Farahani, A., Ahn, J.H., Morrow, K., Kim, N.S.: NDA: Near-DRAM Acceleration Architecture Leveraging Commodity DRAM Devices and Standard Memory Modules. In: Proc. of the Int'l Symp. on High Performance Computer Architecture (HPCA) (2015)
8. Islam, M., Ščrbak, M., Kavi, K.M., Ignatowski, M., Jayasena, N.: Improving Node-Level MapReduce Performance Using Processing-in-Memory Technologies. In: Proc. of the Int'l European Conf. on Parallel Processing (EuroPar) (2014)
9. Joint Electron Devices Engineering Council: High Bandwidth Memory (HBM) DRAM. JEDEC Document #JESD235A (2015)
10. Krishnan, G., Bouvier, D., Zhang, L., Dongara, P.: Energy Efficient Graphics and Multimedia in 28nm Carrizo APU. Presented at Hot Chips (2015)
11. Lee, J., Sathisha, V., Schulte, M., Compton, K., Kim, N.S.: Improving Throughput of Power-Constrained GPUs Using Dynamic Voltage/Frequency and Core Scaling. In: Proc. of the Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT) (2011)
12. Majumdar, A., Wu, G., Dev, K., Greathouse, J.L., Paul, I., Huang, W., Venugopal, A.K., Piga, L., Freitag, C., Puthoor, S.: A Taxonomy of GPGPU Performance Scaling. In: Proc. of the IEEE Int'l Symp. on Workload Characterization (IISWC) (2015)
13. Nair, R., Antao, S.F., Bertolli, C., Bose, P., Brunheroto, J.R.: Active Memory Cube: A Processing-in-Memory Architecture for Exascale Systems. IBM Journal of Research and Development 59(2/3), 17:1–17:14 (March-May 2015)
14. Nowatzki, T., Menon, J., Ho, C.H., Sankaralingam, K.: gem5, GPGPUSim, McPAT, GPUWattch, "Your favorite simulator here" Considered Harmful. In: Workshop on Duplicating, Deconstructing, and Debunking (2014)
15. Paul, I., Manne, S., Arora, M., Bircher, W.L., Yalamanchili, S.: Cooperative Boosting: Needy Versus Greedy Power Management. In: Proc. of the Int'l Symp. on Computer Architecture (ISCA) (2013)
16. Pawlowski, J.T.: Hybrid Memory Cube (HMC). In: Presented at Hot Chips (2011)
17. Pugsley, S.H., Jestes, J., Zhang, H., Balasubramonian, R., Srinivasan, V., Buyuktosunoglu, A., Davis, A., Li, F.: NDC: Analyzing the Impact of 3D-stacked Memory+Logic Devices on MapReduce Workloads. In: Proc. of the Int'l Symp. on Performance Analysis of Systems and Software (ISPASS) (2014)
18. Schulte, M.J., Ignatowski, M., Loh, G.H., Beckmann, B.M., Brantley, W.C., Gurumurthi, S., Jayasena, N., Paul, I., Reinhardt, S.K., Rodgers, G.: Achieving Exascale Capabilities Through Heterogeneous Computing. IEEE Micro 35(4), 26–36 (2015)
19. Su, B., Gu, J., Shen, L., Huang, W., Greathouse, J.L., Wang, Z.: PPEP: Online Performance, Power, and Energy Prediction Framework and DVFS Space Exploration. In: Proc. of the Int'l Symp. on Microarchitecture (MICRO) (2014)
20. TOP 500 List: Titan - Cray XK7. https://www.top500.org/system/177975 (2012), [Online; accessed July-31-2016]
21. Ščrbak, M., Islam, M., Kavi, K., Ignatowski, M., Jayasena, N.: Processing-in-Memory: Exploring the Design Space. In: Proc. of the Int'l Conf. on Architecture of Computing Systems (ARCS) (2015)
22. Wu, G., Greathouse, J.L., Lyashevsky, A., Jayasena, N., Chiou, D.: GPGPU Performance and Power Estimation Using Machine Learning. In: Proc. of the Int'l Symp. on High Performance Computer Architecture (HPCA) (2015)
23. Zhang, D., Jayasena, N., Lyashevsky, A., Greathouse, J.L., Xu, L., Ignatowski, M.: TOP-PIM: Throughput-Oriented Programmable Processing in Memory. In: Proc. Int'l Symp. on High-performance Parallel and Distributed Computing (HPDC) (2014)