IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 10, 2020, accepted March 23, 2020, date of publication March 26, 2020, date of current version April 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2983432

Making Better Use of Processing-in-Memory Through Potential-Based Task Offloading

BYOUNG-HAK KIM AND CHAE EUN RHEE
Department of Information and Communication Engineering, Inha University, Incheon 22212, South Korea

Corresponding author: Chae Eun Rhee (chae.rhee@inha.ac.kr)

This work was supported in part by the R&D Program of MOTIE/KEIT and the NLR Program of MOST/KOSEF under Grant 10077609, in part by the Developing Processor-Memory-Storage Integrated Architecture for Low Power, in part by the High Performance Big Data Servers, and in part by the Basic Science Research Program Through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning under Grant NRF-2018R1A1A1A05023598.

ABSTRACT There is an increasing demand for a novel computing structure for data-intensive applications such as artificial intelligence and virtual reality. The processing-in-memory (PIM) is a promising alternative to reduce the overhead caused by data movement. Many studies have been conducted on the utilization of the PIM taking advantage of the bandwidth increased by the through silicon via (TSV). One approach is to design an optimized PIM architecture for a specific application, the other is to find the tasks that will be more advantageous when offloading to PIM. The goal of this paper is to make the PIM, a newly introduced technology, be easily applied to various applications. The programmable GPU-based PIM is the target system. The essential but simple task offloading conditions are proposed to secure as many candidate tasks as possible when there is any potential benefit from the PIM. The PIM design options then are explored reflecting the characteristics of the candidate tasks actively. When determining offloading conditions, it is difficult to simultaneously consider three time-energy-power objectives. Thus, the problem is divided into two sub-problems. The first offloading condition is designed based on time-energy constraints, whereas the second offloading condition is modeled to satisfy time-power constraints. During the whole processes, the offloading conditions and the PIM design options are carefully configured in a complementary manner to reduce the tasks that are excluded from the offloading. In the simulation results, the suitability of the modeled two offloading conditions and the proposed PIM design are verified using various benchmarks and then, they are compared with previous works in terms of processing speed and energy.

INDEX TERMS High bandwidth memory, near-data-processing, processing-in-memory, task offloading.

I. INTRODUCTION

Recently, more and more attention is paid to applications that require massive data processing, such as artificial intelligence and virtual reality. The parallelism of the graphic processing unit (GPU) has been used to increase the processing speed for such applications. However, improving efficiency in terms of data movement has not been a big concern [1]. Therefore, there is an increasing demand for a novel computing structure optimized to the execution of data-intensive applications. The near-data-processing (NDP), which puts the processing unit close to the data, is a promising alternative for reducing the overhead caused by data movement. When the processor is near memory, it is defined as a processing-in-memory (PIM). An example is the packaging of a processing unit within a memory module or inside a DRAM chip. Recently, the technology of the through silicon via (TSV) enables CPU, GPU or hardware accelerator to be mounted on logic die of 3-dimensional stacked memory. Thanks to this, PIM technology, which vertically stacks DRAM and a logic die with processing units, is also gaining popularity. The programmable GPU-based PIM has been actively studied, because it can process more various algorithms than hardware accelerators. The GPU-based PIM is also attractive, because it allows a use of numerous software development kits (SDK) such as CUDA and OpenCL. Since a memory can be directly accessed through a TSV in the PIM, it is expected that many problems that a host processor has with off-chip memory accesses may be resolved, including the limited bandwidth, long latency and energy inefficiency [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Cihun-Siyong Gong.

The PIM, located at the base layer of the 3D stacked memory, has several challenges to be tackled for the realization and utilization. First, the PIM has a strict area constraint, which makes it difficult to put a sufficient amount of processing units on the logic die of the base layer. In addition, as the base layer of the stacked memory is far off from the top heat sink and thus, it is vulnerable to a thermal problem. This allows just a small power budget for PIM. Under the these design constraints, many studies have been conducted on the utilization of the PIM. The most common approach is to put a PIM simply as a 'mini' version of the host to benefit from the bandwidth increased by the TSV. When the applications are expected to have a higher speed in PIM than in the host processor, job offloading to the PIM is performed with various granularities based on the static or dynamic characteristics of individual applications [1], [3]–[6]. The execution at the compact and low power PIM is expected to result in a high energy-efficiency. Another approach is to design a PIM architecture targeting a specific application such as image processing, machine learning and graphic processing. An optimized hardware accelerator may be proposed [7]–[11]. For the GPU system, the conventional structure or scheduling schemes can be modified [1], [12], [13]. However, these previous studies are not useful enough to drive the popularity of PIM. Strict offloading conditions and PIM design only for specific applications make it difficult for various tasks to be performed in PIM. To make the PIM, a newly introduced technology, be easily applied to various applications, the offloading conditions and the design options should be comprehensively observed, and both need to be complementarily optimized.

The goal of this paper is to increase the applicability of the PIM, taking the processing time, energy and power consumption into consideration in a PIM-enabled GPU consisting of a host GPU and a PIM GPU (Hereafter, the Host GPU and the PIM GPU are referred to as Host and PIM, respectively). The essential but simple task offloading conditions are proposed to secure as many candidate tasks as possible when there is any potential benefit from the PIM. The proposed conditions aim at building up an easy-to-use PIM environment rather than finding the optimal offloading point. The PIM design options then are explored focusing on the candidate tasks. When determining offloading conditions, it is difficult to simultaneously consider three time-energy-power objectives. Therefore, it is divided into two sub-problems of time-energy and time-power. The first offloading condition selects candidate tasks that have the potential to benefit from PIM in terms of energy. PIM design exploration targeting for candidates achieves Host-level processing time with limited computing capacity of PIM. The second offloading condition takes care of the power constraint of PIM. Through the clock frequency scaling technique, the possibility of candidate tasks falling out of PIM offloading is reduced. The main contributions of this paper are summarized as follows.

- Simplified offloading conditions are proposed supported by PIM design option exploration to make better use of PIM through potential-based task offloading.
- Design challenges of processing speed, energy efficiency and power budget are considered together.
- Setup and use of PIM environment are easy with a few intuitive parameters.

The remainder of this paper is organized as follows. Section II reviews the previous works about PIM offloading. In Section III, an overview of the proposed PIM-enabled GPU system is given. Sections IV and V respectively propose the task offloading conditions and the PIM design in view of the processing time and energy efficiency. Section VI describes the task selection process to satisfy the power constraints. Section VII shows simulation results and the conclusions are given in Section VIII.

II. RELATED WORKS

Many studies have already pointed out the memory bottleneck problem and highlighted the need to shift the paradigm to data centric computing [14], [15]. They have shown that the consideration of PIM-specific constraints, a new architecture for the PIM, the set of memory intensive benchmarks to benefit from the PIM and the methods to accurately identify the PIM offloading candidates are necessary. Also, the simulation infrastructure to measure the performance of the PIM should be established [14], [15].

In most of the recent studies, out of the many tasks constituting an application, those that require a large amount of memory bandwidth are offloaded to the PIM [1], [5]. In [1], the offloading tasks are selected by three conditions of memory intensity, shared memory intensity and available parallelism. After that, a precise scheduling algorithm is applied. In [5], the performance after the offloading is predicted based on the number of the executed vector and scalar commands, the number of memory access commands, the number of the local data shared accesses, the utilization rate of various execution units, the cache hit rate and the amount of memory access. In these previous studies, the offloading conditions are too complicated and various to be practically used. Strict and complex offloading conditions are difficult to use and lower the utilization of PIM by reducing the chance that tasks will run in PIM.

Some studies have focused on the power constraints of the PIM to consider the thermal problem in addition to processing speed-up [16]–[19]. Various solutions from passive cooling to high-end server active cooling have been applied to demonstrate the feasibility of the PIM with a small power budget [16]. Reference [17] improves the speed under the thermal design power limit by designing SW/HW that can recognize heat during run time and offload instructions to PIM considering thermal issue in PIM. In [18], an architecture has been proposed to optimize the power efficiency of PIM. A network on chip of diagonally linked mesh (DMesh), which is proposed instead of the conventional fully connected

crossbar, occupies less area. In a saved space, additional SMs are invested. It greatly increases energy efficiency satisfying the power constraints. The [19] analyzed the effect of dynamic voltage and frequency scaling (DVFS) on the performance of the Host and PIM. Tasks are offloaded to the PIM according to four scenarios of maximum performance, minimum power, minimum energy delay squared product (ED2P) and maximum performance under power constraint. However, these studies [18], [19] focus on reducing the power consumption of PIM, which can reduce processing speed.

Some works have studied the architecture of GPU-based PIM for specific area such as image or graph processing [12], [13]. The work in [12] proposed a PIM architecture and programming model for image processing. Compact GPU design is achieved by efficiently distributing data to each SM based on a regular memory access pattern. Reference [13] designed an HMC-based graphic processor architecture for 3D rendering applications. The texture filtering performance is significantly improved by applying the frame tiling technique to the texture unit in HMC. An architecture optimized for a specific task has the disadvantage of lack of design flexibility or applicability.

III. OVERVIEW OF THE PROPOSED PIM-ENABLED SYSTEM

The GPU-based PIM is one of the architectures that draw most attention. The PIM enable system in this paper consists of the Host and the PIM, as assumed in many previous studies [1], [5], [13], [18]–[20]. The Host comprises tens or hundreds of streaming multiprocessors (SM), which are connected to the HBM through interposers. The PIM is mounted on a logic die placed in the base layer of the HBM. Due to the limitations of the area, the PIM has a small number of SMs, and thus the computing capability is considerably lower than that of the Host. However, the PIM may have a high memory bandwidth. Unlike HMC, the current HBM has same external and internal memory bandwidth. However, many current studies, including [1], assume that higher internal BWs can be utilized when the processor is mounted on logic dies of 3D stacked memory such as HBM. This is because PIM is connected to the memory controller through TSV, so there is no need for unnecessary external interface. This leads to various possibilities to increase internal memory bandwidth in the near future. In the proposed PIM-enabled system, 3D stacked memory with HBM configuration is assumed.

Figure 1 shows the development process of the proposed PIM-enabled system. It consists of the process of determining offloading conditions and the PIM design exploration to support those conditions after test running. Since it is difficult to acquire the sufficient information for offloading decision during the compile time, Host collects the performance counter and power information through a test running. In the first stage, the $OC_{T\text{-}E}$ is modeled for task selection with energy advantages in PIM. The opportunity to be a candidate task is given, assuming that the tasks having a difficulty in the memory access in the Host may benefit from the high

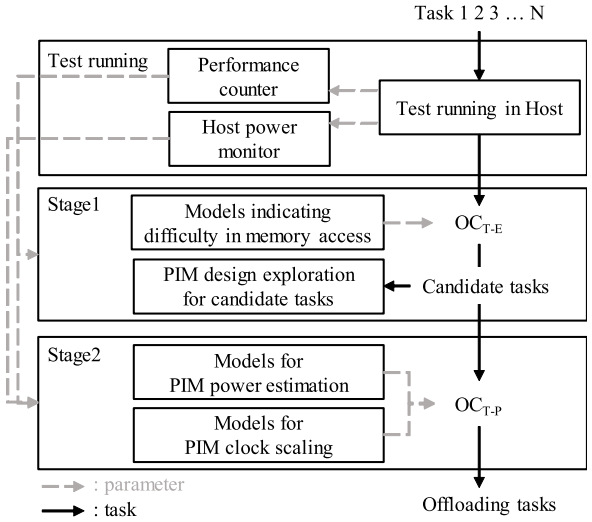

FIGURE 1. development process of the proposed system.

memory bandwidth of the PIM. Then, PIM design exploration is conducted reflecting the characteristics of the candidate task selected by the $OC_{T\text{-}E}$ so that PIM running can be competitive in terms of processing speed compared to Host running.

In the second stage, the offloading condition, OC_{T-P}, is modeled to satisfy the PIM power constraint. Due to the physical position, the PIM gives a direct effect on the memory temperature. If the power budget allowed to the PIM is exceeded and the temperature is increased, the refreshing period of the memory will be shortened. This can significantly decrease the performance of the entire system because the memory is shared by the PIM and the Host. However, if the candidate tasks that exceed the power budget are indiscriminately excluded from the PIM offloading, the applicability of the PIM that may benefit from the high energy efficiency will be decreased. Operating clock scaling techniques enable power consumption control of candidate tasks. Test running in Host enables the estimation of power consumption and processing speed in PIM according to operating clock scaling.

The two offloading conditions and the power estimation for PIM are designed by means of straightforward and easy-to-use linear combination model. The parameters used in modeling are intuitive and very small compared to previous studies. Once the thresholds and weights of $OC_{T\text{-}E}$ and OC_{T-P} are set according to the user's system following the development process of Figure 1, only offloading condition checking is required during the use process. Just marginal time overhead is required to calculate $OC_{T\text{-}E}$ and OC_{T-P}. The accuracy of the models proposed in this paper is experimentally verified in Section VII.

IV. POTENTIAL-BASED OFFLOADING CONDITIONS FOR ENERGY EFFICIENCY

In the Host environment with many SMs, the high parallel processing may easily improve the task execution speed but

it significantly decreases the cache efficiency. The high miss rate of the cache increases the access to a lower level cache or an off-chip memory, often causing the stall or the lack of bandwidth. Considering such characteristics, the miss per total instruction of the L2 cache (*L2MPI*) and the stall of the L1 cache (*L1S*) are selected as the indicators to check the potential of specific tasks. As the *L2MPI*, representing the frequency of miss in the L2 cache, is increased, the access to the off-chip memory is increased. In [21], tasks with high misses per kilo instructions showed higher processing speed and lower energy-delay-product in PIM compared to Host. In that case, it is advantageous to execute the tasks in a PIM having a high memory bandwidth. The *L1S* represents the stall from the L1 cache to the L2 cache [22]. The tasks that frequently undergo *L1S* have the potential to better utilize the high memory bandwidth of the PIM when the stall is decreased. The $OC_{T\text{-}E}$ in (1) checks whether the weighted summation of the *L1S* and *L2MPI* is greater than the threshold TH. The weight values and the TH are set empirically through experiments. If the $OC_{T\text{-}E}$ value of a particular task that is eligible for PIM is less than the threshold, the task may miss the opportunity to be offloaded to the PIM. The threshold is therefore set to a sufficiently small value with a margin. However, if the $OC_{T\text{-}E}$ value of which a task that is not suitable for PIM is rather high, it may be determined as a candidate task inappropriately. Wrong offloading decision will result in poor performance. In this paper, memory-access-oriented PIM design exploration is performed considering the characteristics of candidate tasks as well as offloading conditions. This greatly makes the memory access smooth in PIM, allowing candidate tasks to achieve higher processing speed compared to Host. As such, $OC_{T\text{-}E}$ and PIM design exploration are complementary, reducing the risk of setting the threshold to a small value. The weight values in (1) are determined to balance different ranges of *L2MPI* and L *L1S* 1S values. For the 17 tasks, the average of *L2MPI* and *L1S* was 0.003 and 0.17, respectively, and thus, a weight of 58(=0.17/0.003) is given for *L2MPI*. For *L1S*, a weight of 0.22 is set to increase the importance of *L2MPI*, which directly affects the utilization of the wide memory bandwidth of PIM.

$$58 \times L2MPI + 0.22 \times L1S + 0.005 > TH \tag{1}$$

To test the suitability of the $OC_{T\text{-}E}$, various benchmarks are executed in the Host and PIM environments. Seventeen benchmarks from rodinia3.0 [23], Poly-Bench [24] and NVIDIA SDK CUDA code samples [25] are run in the gpgpu-sim [22]. The baseline system configuration follows Pattnaik's work [1]. A GPU having a Fermi architecture is assumed. For 3D stacked memory configuration, the dram latency, bus width, burst length and clock frequency of GDDR5 are scaled based on [26] and the HBM timing model of Gem5 [27] is used. To clearly discern the effect of the memory bandwidth and computing performance, the PIM is configured to have 1/4 computing capability and 4 times memory bandwidth compared to the Host. The number of SMs is 32 and 8 in the Host and the PIM, respectively. The 3D stacked memory bandwidth is assumed to be 256 GB/s in the PIM and 64 GB/s in the Host. In both Host and PIM, the L1 data cache size is 16 KB, whereas the L2 cache size is 128 KB per memory partition, and 1 MB in total for all the eight memory partitions. The operating clock frequency is set to be 1 GHz for both Host and PIM. Hereafter, the PIM with this configuration is called a PIM_{BASE}.

Table 1 shows whether the $OC_{T\text{-}E}$ can be used to determine the competitiveness of PIM. The first column shows the benchmarks used for the experiment, and the second column shows the calculated value of the left-hand in (1). The third column shows the instructions per cycle (IPC) ratio obtained by executing the same benchmarks in the Host and PIM_{BASE}. At TH=0.1, the shaded twelve tasks satisfy the $OC_{T\text{-}E}$. Among them, six have a higher IPC in the PIM_{BASE} than in the Host. Potential does not always guarantee higher PIM IPC than Host. Considering that SM accounts for about 80% of the entire power consumption in the GPU [28], the power consumption of PIM is overwhelmingly low compared to Host. If task offloading is immediately given up, the opportunities for energy saving would disappear. In the following section, the IPC of PIM is improved to the level of the Host by positively reflecting the characteristics of the candidate tasks to the PIM design options.

V. PIM DESIGN EXPLORATION FOR ENERGY EFFICIENCY

In Table 1 of Section IV, CS, J2I and Conv2 satisfy the $OC_{T\text{-}E}$ but actually show a lower IPC in PIM_{BASE} than in Host. Focusing on these target tasks, the PIM design exploration is conducted to make the potential-based $OC_{T\text{-}E}$ valid. To increase the IPC of the target tasks in PIM_{BASE} to the level of Host, the speed improvement by about 56% is required.

TABLE 1. The relationship between $OC_{T\text{-}E}$ and PIM-to-Host IPC Ratio.

Benchmarks	$OC_{T\text{-}E}$	Host-to-PIM IPC ratio
Bfs	1.89	1.44
Hybridsort (HS)	0.49	1.40
Reduction (RC)	0.29	1.03
StreamCluster (SC)	0.20	1.04
ScalarProd (SP)	0.18	1.08
Dwt2d	0.15	0.83
Jacobi-2d-imper (J2I)	0.14	0.73
ConvolutionSeparable (CS)	0.19	0.83
Sradv2	0.11	0.45
FastWalshTransform (FWT)	0.11	1.08
Convolution-2d (Conv2)	0.10	0.35
Sradv1	0.10	0.70
Histogram (HG)	0.04	0.34
Pathfinder (PF)	0.03	0.28
LavaMD	0.02	0.63
BinomialOptions (BO)	0.005	0.25
Jacobi-1d-imper (J1I)	0.005	0.52

A. SM

During memory-intensive workloads, increasing the number of computation cores does not guarantee a performance boost

at any point due to the limited memory bandwidth [29]. If the number of computation cores increases in data-demanding tasks aimlessly, the performance may even become degraded due to a large amount of idle state cores caused by a long memory access latency time. However, in an environment where the number of SMs is very low and the memory bandwidth is sufficient, such as PIM, additional SMs are likely to contribute directly to a performance improvement. Because the SM is the most important factor affecting the area of a GPU, it is necessary to select an appropriate number of SMs considering both the IPC and area overhead. Figure 2 shows the candidate task's IPC based on the number of SMs when the memory bandwidth is 64 and 256 GB/s. When the memory bandwidth is small like the Host, IPC increases and soon saturates due to the memory bottleneck as shown in CS and J2I. However, when high memory bandwidth such as PIM, a proportional improvement in IPC is observed as the number of SMs increases. Conv2 has a lot of memory access, but the L1 cache is the main bottleneck. Thus, off-chip memory bandwidth does not significantly affect performance. The 3D stacked memory chip size is known to be 91.99 mm^2, and the free space is about 72.25 mm^2, if the area for physical layer (PHY), TSV and test ports (19.74 mm^2) is excluded [30]. The area for single SM, an L2 cache and a memory controller is approximately 5.4 mm^2, 3.8 mm^2 and 2.04 mm^2, respectively. Thus, the total area of PIM_{BASE} comes to 49.04 mm^2. Therefore, up to 12 SMs can be mounted within the PIM's area budget. The area of each component is estimated according to CACTI7 [31] and MCPAT [32]. When the target tasks are executed in the PIM with 12 SMs, denoted as PIM_{SM12}, the IPC is improved by about 48% compared to the PIM_{BASE} with eight SMs. The IPC needs to be increased 8% more to have a processing speed similar to that of Host. Therefore, better design options are necessary for caches.

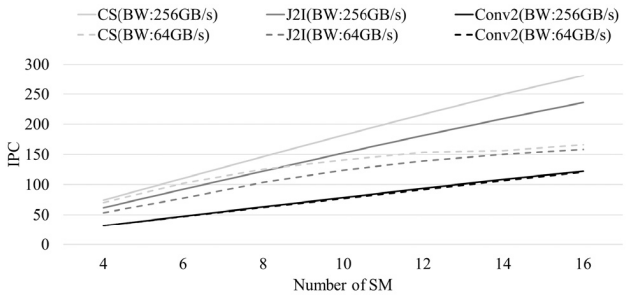

FIGURE 2. Comparison of IPC values according to the number of SMs.

B. CACHE

Massive multi-threading of the GPU lowers the effective cache size per thread, causing significant cache conflicts [33]. Even if the size of off-chip memory is assumed to be infinitely large, the IPC performance is limited by the cache contention [34]. Therefore, there is an opportunity to increase IPC if the cause of L1 and L2 cache contention is identified and resolved well. To analyze cache contention, the reservation fail (RF) [35] is checked. The RF represents the contention of the resource handling for cache miss and is classified into four types. The *RF_Tag* means that a cache line is reserved by a previous memory request and fails to be replaced, whereas the *RF_Miss_queue* represents that a miss request is failed to be sent to a lower memory level as the buffer is fully occupied by the bandwidth limitation. If there is a cache miss of the same data in the miss status holding register (MSHR), the pending hit is increased after merging. The *RF_Merge_MSHR* means that the number of merges allowed in one MSHR has been exceeded. Finally, the *RF_Scarce_MSHR* occurs when a miss request fails to be registered due to the lack of the entry in the MSHR. This subsection explores a cache design option that reduces the RF observed to be high in cache and consequently decreases the memory access latency. In gpgpu-sim, the latency is modeled through queues between each memory. Among them, the maximum round trip time between cores and DRAM is observed through the sum of *max_icnt2mem_latency* and *max_icnt2sh_latency*.

1) L1 CACHE

The L1 cache miss latency plays an essential role for the IPC of the memory intensive application [34]. In the PIM_{SM12} of Figure 3, *RF_Tags* of CS, J2I and Conv2 account for 98.4%, 99.9% and 99.9% in total RF, respectively. Thus, the L1 cache miss latency can be predicted by the *RF_Tag*. As mentioned above, this RF indicates the lack of a replaceable cache line. When the number of ways of the L1 data cache is doubled to reduce the conflict, not only does the *RF_Tag* value decrease but also the total RF value is lowered as shown in PIM_{SM12_L1} of Figure 3. In this case, the number of sets is halved to maintain the cache size. As a result, the maximum round trip latency between SM and DRAM is reduced by an average of 71%. The PIM_{SM12} with this type of L1 cache is denoted by PIM_{SM12_L1}, hereafter.

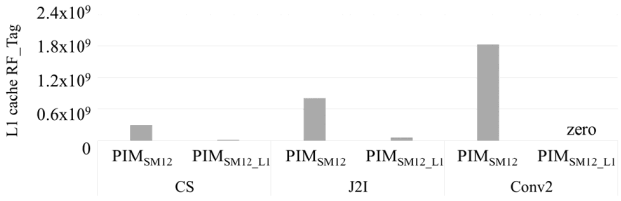

FIGURE 3. Comparison of the RF_TAG of the L1 cache.

2) L2 CACHE

In conventional GPU environment, L2 cache is known to have a lot of congestion because it receives requests from both off-chip memory and L1 cache [34]. This tendency is worse when the memory bandwidth is larger. Therefore, for applications with high data demands, it is common to increase the L2 cache size along with the number of SMs. L2 cache contention will be very low when very few SMs are used, such as the PIM_{BASE}. However, in the case of PIM_{SM12_L1}, the number of SMs is slightly increased, and above all, the data traffic flow to the L2 cache is significantly smoothed

due to the structure change of the L1 cache. This may increase L2 cache contention [34]. In fact, the *RF_Miss_queue* of L1 cache in PIM_{SM12_L1} increased significantly by 2 times for CS and 400 times for J2I compared to PIM_{SM12}. This is because the miss request is not sent from the L1 cache due to the contention of the L2 cache.

In Table 2, $PIM_{RPROPSED}$ is a configuration that doubles the number of MSHR entry and the maximum number of merges of the L2 cache compared to PIM_{SM12_L1}. The number of MSHR entries and the maximum number of merges for PIM_{SM12_L1} were 32 and 4, respectively, and increased to 64 and 8, respectively. Also, the area of $PIM_{PROPOSED}$, which has doubled the MSHR entry, is 70.67 mm^2, meeting the area budget of PIM. The RF distribution of L2 cache is compared for three target tasks. The *RF_Miss_queue*, *RF Tag* and *RF fail* are not given in Table 2 because the memory bandwidth of the PIM is set to a large value of 256 GB/s and thus, the probability of those RF is very rare. In PIM_{SM12_L1}, it is observed that *RF_Scarce_MSHR* and *RF_Merge_MSHR* of CS are very high. In $PIM_{RPROPSED}$, *RF_Scarce_MSHR* and *RF_Merge_MSHR* of CS is decreased by 89% and 20%, respectively, and overall L2 cache RF is reduced by 35%. As a result, for CS, the round-trip latency between SM and DRAM is reduced by 20%.

TABLE 2. Comparison of the RF_Merge, RF_Scarce MSHR of the L2 cache.

		L2 RF Merge MSHR	L2 RF Scarce MSHR
CS	PIM_{SM12_L1}	48105	115223
	$PIM_{PROPOSED}$	38866	12729
J2I	PIM_{SM12_L1}	334	0
	$PIM_{PROPOSED}$	224	0
Conv2	PIM_{SM12_L1}	280	0
	$PIM_{PROPOSED}$	169	0

VI. REFINED OFFLOADING CONDITIONS FOR POWER CONSTRAINTS

The power budget of the PIM depends on the cooling system. When using high-end-server active heat sink, it is known as 55W [16], but current power consumption of the Host is much higher. In this section, the power consumed by candidate tasks in the $PIM_{PROPOSED}$ is modeled based on running tests on the Host. If their $Power_{PIM}$ exceeds the allowed power budget, OC_{T-P} is adopted to give them a second chance for offloading. By lowering the operating clock frequency, it is checked whether the PIM execution time, $Time_{PIM\_Scaled}$, can still maintain the level of Host execution time, $Time_{Host}$, under the $Power_{PIM\_Scaled}$ less than the power budget. To do this, both $Time_{PIM\_Scaled}$ and $Power_{PIM\_Scaled}$ need to be modeled from the test running of Host.

A. POWER ESTIMATION FOR THE PROPOSED PIM

Since static power is proportional to the area of active components, it can be roughly estimated by reflecting the difference of the number of SMs and cache size between the Host and the $PIM_{PROPOSED}$. However, dynamic power is more affected by the characteristics of the task than the GPU structure. In addition, the dynamic power consumption ratio of the Host and $PIM_{PROPOSED}$ may be different for each SM, L2 cache, interconnection network (ICNT), off-chip memory DRAM and memory controller (MC) components. Therefore, independent power estimation is required for each component. In the PIM environment with a small number of SMs, the idle SM is very rare and thus, only the power change of the active SM is considered.

Two steps are conducted in this paper to estimate the power of $PIM_{PROPOSED}$. First, the dynamic power ratio between PIM_{SM12} and Host is modeled reflecting the SM number difference. The model is then adjusted to take into account the changed cache design options. For the power model of PIM_{SM12} in (2), the GPU components are divided into two parts based on the experimentally obtained P_{SM12}-to-H ratio. GPU component parts with similar P_{SM12}-to-H ratio values have the same weight. Due to the high memory bandwidth, the processing utilization rises, resulting in an overall increase in the power of the computational part. In the case of L2 cache, the size is large enough and it is shared by all SMs and thus, the miss rate change according to the number of SMs is not large. Thus, it is observed that, in many target tasks, dividing P_{SM12}-to-H of DRAM by P_{SM12}-to-H of active SM is about 0.25. From this, the memory part consisting of MC and DRAM has one quarter the weight of the computational part.

$$\begin{aligned} &PIM_{SM12} \\ &\quad = \alpha \times ((\text{ActiveSM\_Host} + \text{L2\_Host} + \text{ICNT\_Host}) \\ &\quad +0.25 \times (\text{MC\_Host} + \text{DRAM\_Host})) \end{aligned} \tag{2}$$

The dynamic power of $PIM_{PROPOSED}$ is obtained by adjusting the model of (2) using the β value as shown in (3). The power in L2 cache and ICNT is minorly tuned with obtained constant values.

$$\begin{aligned} &PIM_{PROPOSED} \\ &\quad = \beta \times ((\text{ActiveSM\_}PIM_{SM12} + \text{MC\_}PIM_{SM12} \\ &\quad +\text{DRAM\_}PIM_{SM12}) \\ &\quad +0.8 \times (\text{ICNT\_}PIM_{SM12} + \text{L2\_}PIM_{SM12})) \end{aligned} \tag{3}$$

In order to utilize the model (2) for a particular task, α needs to be known, which is obtained using Idle per active (*IPA*) and *L2MPI* collected from the Host's test running. If the Host's *L2MPI* is small, there will be little change in power because it will not benefit from the increased memory bandwidth of the PIM. Otherwise, processing utilization is improved, which consumes more power than Host. *IPA* is the ratio of the number of active SMs to idle SMs. The large *IPA* in the Host means that this task can be executed with only a small number of active SMs. Thus, the power consumption will not change much when running in PIM. In the first column of Table 3, the α value is determined by dividing

TABLE 3. Alpha depending on L2MPI and IPA.

L2MPI/coeff_1	α	IPA/coeff_2	Reduction rate of α (%)
1 ≤	1.8	1 ≤	0
[0.4,1)	1.6	[0.01, 1)	10
[0.2,0.4)	1.4	[0.0001, 0.01)	20
< 0.2	1	< 0.0001	30

L2MPI by coeff_1. The α is then finely adjusted according to the *IPA* divided by coeff_2. Coefficients and the range of α are empirically derived through experiments of twelve candidate tasks, where coeff_1 and coeff_2 are set to 0.0055 and 0.38, respectively.

In estimating the dynamic power of $\text{PIM}_{\text{PROPOSED}}$ in (3), β is set using L1 cache *RF_Tag* divided by total instructions (*L1R_TPI*) from the Host's test running. Coefficients and the range of β are empirically derived through experiments of twelve candidate tasks, where coeff_3 is set to 0.09. ICNT and L2 cache have a weight of $\beta \times$ 0.8. This is obtained by dividing the average $\text{PIM}_{\text{PROPOSED}}$-to-$\text{PIM}_{\text{SM12}}$ power ratio of the L2 cache by the average $\text{PIM}_{\text{PROPOSED}}$-to-$\text{PIM}_{\text{SM12}}$ power ratio of the active SM.

Table 4 is derived from the correlation between the power ratio of $\text{PIM}_{\text{PROPOSED}}$ to PIM_{SM12} ($\text{P}_{\text{PROPOSED}}$-to- P_{SM12}) and *L1R_TPI* in the Host's test running. It is based on that the reduction of *L1R_TPI* from PIM_{SM12} to $\text{PIM}_{\text{PROPOSED}}$ shows a similar tendency with *L1R_TPI* of Host. In Figure 4, the left vertical axis represents the power ratio, whereas the right vertical axis represents *L1R_TPI*. A dashed graph means the reduction of *L1R_TPI* when changing from PIM_{SM12} to $\text{PIM}_{\text{PROPOSED}}$. If this value is small, the task's $\text{P}_{\text{PROPOSED}}$-to-$\text{P}_{\text{SM12}}$ power ratio = 1, showing little change in power. Otherwise, it is observed that the power ratio gets high because the processing utilization of each component increases as the contention of the L1 cache decreases. Meanwhile, Host *L1R_TPI* marked by solid line and the reduction of *L1R_TPI*

TABLE 4. Beta depending on L1R_TPI.

L1R_TPI /coeff_3	β
1 ≤	1.2
[0.5, 1)	1.1
< 0.5	1

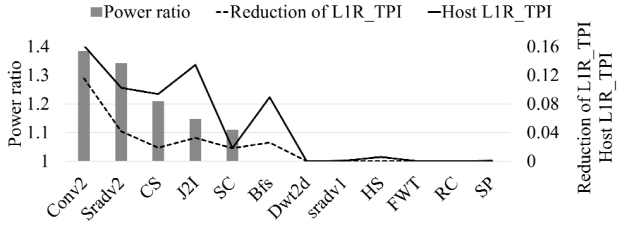

FIGURE 4. $\text{PIM}_{\text{PROPOSED}}$ to PIM_{SM12} power ratio depending on L1R_TPI of Host.

also show a roughly proportional relationship. Tasks with high *L1R_TPI* in the Host show the same characteristics in PIM, too. Due to improved processing utilization in $\text{PIM}_{\text{PROPOSED}}$ with cache modification, the *L1R_TPI* reduction of these tasks is also increased.

B. REFINED OFFLOADING CONDITIONS

In order to apply the $\text{OC}_{\text{T-P}}$ to candidate tasks whose $\text{Power}_{\text{PIM}}$ estimated in Section IV.A is greater than the allowed power budget, the amount of downscaling in operating clock frequency to satisfy the power budget should be known. The corresponding $\text{Power}_{\text{PIM\_Scaled}}$ and $\text{Time}_{\text{PIM\_Scaled}}$ are also necessary.

In Figure 5, for $\text{Power}_{\text{PIM\_Scaled}}$ estimation, the change of total power consumption including static and dynamic according to operating clock frequency is observed for candidate tasks. The power value is normalized to $\text{Power}_{\text{PIM}}$ at the operating clock frequency of 1000MHz. When it is downscaled to 900, 800, 700 and 600Mhz, the power reduction rate compared to $\text{Power}_{\text{PIM}}$ is about 5.3%, 11%, 17%, and 24.4%, respectively. The standard deviation among tasks is not large, around 2%. Through this, the amount of downscaling that satisfies the power budget and the corresponding $\text{Power}_{\text{PIM\_Scaled}}$ information can be estimated.

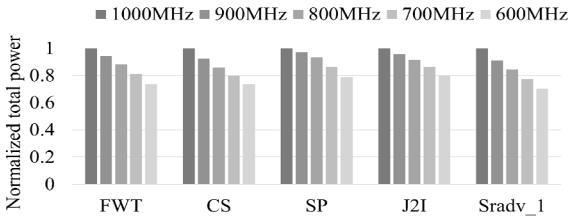

FIGURE 5. The change of total power consumption according to operating clock frequency for target tasks.

For $\text{Time}_{\text{PIM\_Scaled}}$, direct estimation from $\text{Time}_{\text{Host}}$ is very challenging. This paper obtains that indirectly through T_{ratio} (P-to-H) and T_{ratio} (P_{Scaled}-to-P) as shown in (4). In (5), T_{ratio} (P-to-H) is the time ratio when both $\text{PIM}_{\text{PROPOSED}}$ and Host run at the baseline operating clock frequency of 1000 MHz. T_{ratio} (P_{Scaled}-to-P) of (6) shows the ratio between $\text{Time}_{\text{PIM\_Scaled}}$ and Time_{PIM}.

$$\begin{aligned} T_{ratio}(P_{Scaled}-to-H) &= Time_{PIM\_Scaled}/Time_{Host} \\ &= T_{ratio}(P-to-H) \\ &\quad \times T_{ratio}(P_{Scaled}-to-P) \end{aligned} \tag{4}$$

$$T_{ratio}(P-to-H) = Time_{PIM}/Time_{Host} \tag{5}$$

$$T_{ratio}(P_{Scaled}-to-P) = Time_{PIM\_Scaled}/Time_{PIM} \tag{6}$$

To obtain T_{ratio} (P-to-H) in (5), the dynamic power ratio of $\text{PIM}_{\text{PROPOSED}}$ and Host for active SM in Section VI.A is used. This is an important indicator of the processing utilization of tasks in $\text{PIM}_{\text{PROPOSED}}$ versus Host. As shown in Figure 6, the active SM power ratio and measured IPC ratio of $\text{PIM}_{\text{PROPOSED}}$ to Host are almost identical.

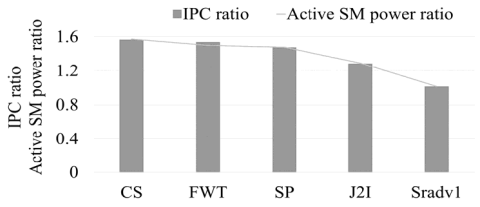

FIGURE 6. In $PIM_{PROPOSED}$ and host, measured IPC ratio and power ratio of active SM.

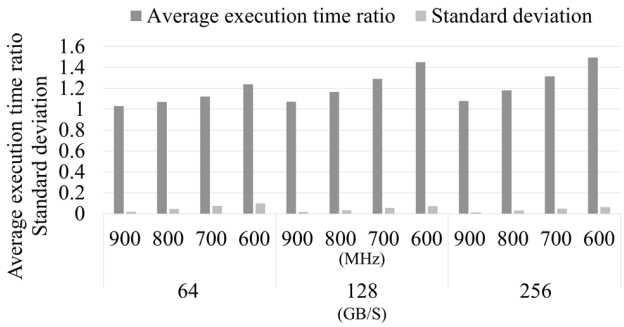

FIGURE 7. Execution time ratio depending on the operating clock frequency in different memory bandwidth.

Figure 7 shows the experiment to obtain T_{ratio} (P_{Scaled}-to-P) of (6). Under the memory bandwidth of 64, 128, and 256 GB/s, the average and standard deviation of the execution time change is illustrated when the operating clock frequency changes from 1000 MHz to 900, 800, 700, and 600 MHz. It is observed that the standard deviation is very small in large memory bandwidth environments. It means that, in $PIM_{PROPOSED}$ with 256GB/s memory bandwidth, the execution time ratio (P_{Scaled}-to-P) can be determined mostly according to the change of operating clock frequency without caring about each task independently. When the memory bandwidth is 256GB/s, T_{ratio} (P_{Scaled}-to-P) gradually increases to 1.07, 1.18, 1.31 and 1.49 when the operating clock frequency is downscaled from 1000 MHz to 900, 800, 700 and 600MHz, respectively. At low operating clock frequencies, the standard deviation is not negligible and thus, there may be tasks that fall outside the estimated time ratio. That is, there is a risk of offloading even when $Time_{PIM\_Scaled}$ is larger than $Time_{Host}$. To avoid this, $Time_{PIM\_Scaled}$ is conservatively estimated from $Time_{PIM}$ by adjusting T_{ratio} (P_{Scaled}-to-P) = 1.1, 1.23, 1.37 and 1.56 when operating clock frequency = 900, 800, 700, 600MHz, respectively.

VII. EXPERIMENTAL EVALUATION

A. SUITABILITY ANALYSIS OF THE OCT-E IN THE PROPOSED PIM

Table 5 is the recalculation of the Host-to-PIM IPC ratio of Table 1 based on the newly measured IPC in the $PIM_{PROPOSED}$. The gray-shaded tasks have an IPC ratio greater than 1, meaning that the execution speed is higher in the $PIM_{PROPOSED}$ than in the Host. Among the twelve candidate tasks that satisfy the $OC_{T\text{-}E} > TH$ (=0.1), only Sradv_2 has an IPC ratio smaller than 1. All the tasks that fail to satisfy the $OC_{T\text{-}E}$ have a smaller IPC compared to Host. From Table 5, a quite large number of tasks become candidates to offload, satisfying the $OC_{T\text{-}E}$ thanks to the modification of the PIM design.

TABLE 5. The relationship between proposed PIM-to-Host IPC and potential based offloading conditions.

Benchmark	$OC_{T\text{-}E}$	Host-to-PIM IPC ratio
Bfs	1.89	1.92
HS	0.49	1.68
RC	0.29	1.37
SC	0.20	1.24
SP	0.18	1.49
Dwt2d	0.15	1.23
J2I	0.14	1.28
CS	0.19	1.57
Sradv2	0.11	0.92
FWT	0.11	1.54
Conv2	0.10	1.05
Sradv1	0.10	1.02
HG	0.04	0.50
PF	0.03	0.42
LavaMD	0.02	0.76
BO	0.005	0.37
J1I	0.005	0.68

B. SUITABILITY ANALYSIS OF THE OCT-P IN THE PROPOSED PIM

The power consumption of $PIM_{PROPOSED}$ and $Time_{PIM\_Scaled}$ estimated in Sections VI.A and B are compared with the actual simulation results. In Table 6, the second column represents the estimated $Power_{PIM}$ of $PIM_{PROPOSED}$, whereas the third column represents the value measured by the simulator. As presented in the fourth column, the average error rate is about 3.62%. Tasks whose $Power_{PIM}$ is over power budget(=55W) are shaded in gray. Their error rate is 5.62%, whereas the other tasks have a smaller error rate of 2.25%. In other words, the task that consume low power is less likely to be excluded from offloading due to power budget

TABLE 6. Comparison of estimated and measured power consumption.

Benchmark	Estimated (W)	Measured (W)	Error rate (%)
Dwt2d	50.14	49.42	1.45
CS	65.38	67.30	2.84
J2I	60.49	59.99	0.83
Sradv_2	48.46	49.82	2.72
Conv2	51.09	52.33	2.36
Bfs	51.58	52.57	1.89
SC	49.68	47.07	5.55
RC	54.53	54.19	0.62
HS	51.62	52.29	1.27
SP	57.14	58.66	2.59
FWT	56.71	63.64	10.89
Sradv_1	63.03	57.07	10.44
Mean	54.99	55.37	3.62

limitation. It meets the purpose of this paper to improve the PIM utilization.

Table 7 compares the estimated $\text{Time}_{\text{PIM\_Scaled}}$ with measured values through simulation. The average error rate is 5.8%. $\text{Time}_{\text{PIM\_Scaled}}$ of Sradv_1 is larger than $\text{Time}_{\text{Host}}$ and is excluded from offloading. Four out of five tasks exceeding the power budget could be offloaded through a second chance.

TABLE 7. Comparison of Estimated and Measured $\text{Time}_{\text{PIM\_Scaled}}$.

Benchmark	Estimated (s)	Measured (s)	Error rate (%)
CS	697×10^{-4}	742×10^{-4}	6.4
J2I	119×10^{-3}	124×10^{-3}	4.3
SP	117×10^{-6}	119×10^{-6}	1.8
FWT	190×10^{-4}	199×10^{-4}	4.7
Sradv_1	220×10^{-3}	194×10^{-3}	11.9
Mean	859×10^{-4}	827×10^{-4}	5.8

C. PERFORMANCE COMPARISON

In this subjection, the contribution of the proposed technique to the performance improvement of the PIM-enabled system over Host-only system is verified compared to techniques in [19]. In addition, performance of the proposed PIM is compared with [18].

First, techniques proposed in [19] and in this paper are verified by comparing execution time and ED2P reduction of PIM-enabled system over Host-only system. The system environment used in [19] and in this paper are not exactly the same, but it is reasonable to compare the performance improvement for the following reasons. In PIM-enabled system of [19], PIM has 0.75 × SMs compared to Host, whereas PIM of this paper has 0.375 × SMs compared to Host. The PIM-to-Host computing capability in [19] is larger than in this paper. In [19], however, a low-performance embedded GPU is used for PIM, and the memory bandwidth of Host is set higher than in this paper. Therefore, it can be said that the system of [19] has a similar PIM-to-Host computing ratio as assumed in the proposed system. In addition, the main technique proposed in [19] is DVFS of which performance is not largely dependent on small structural differences.

Figure 8 shows the reduction of time and ED2P in $\text{PIM}_{\text{PROPOSED}}$ and Marko's PIM [19]. They are normalized based on performance when running in the host environment defined in both studies. Dark and light gray bars represent a normalized time and a normalized ED2P, respectively. When eleven tasks are offloaded to $\text{PIM}_{\text{PROPOSED}}$, time and ED2P are decreased by 20.5% and 67.2%, respectively. Compared to Marko's host, the average reduction rate of time and ED2P of Marko's PIM applied with their DVFS technique is shown in the rightmost. Time and ED2P are reduced by 20.4% and 61%, respectively, for the fifteen tasks. Marko drastically reduces time and ED2P, where the optimized kernel deployment and operating clock frequency adjustment are searched to achieve minimum time or minimum ED2P on a PIM-enabled system with a 14 nm technology. Here, however, the power budget is not considered. This paper achieves

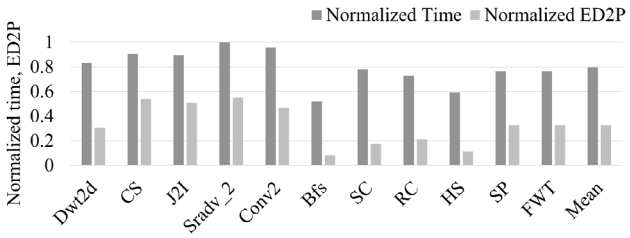

FIGURE 8. Reduction of time and ED2P in PIMPROPOSED and Marko's.

a reduction rate of time and ED2P similar to Marko's, while satisfying the area and power budget constraints in a PIM-enabled system with a 28 nm technology.

Second, IPC and ED2P changes are shown for the PIMs proposed in [18] and in this paper compared to the same baseline PIM, baseline2 of [18]. Simulation is done using same benchmarks. The proposed offloading condition and PIM design options are verified using various tasks not used for modeling. For the twenty kernels used in Wen [18], the $\text{PIM}_{\text{PROPOSED}}$ is compared with Wen's in terms of IPC and ED2P. Table 8 shows the Host to PIM IPC ratio and $\text{OC}_{\text{T-E}}$ for Wen's kernels. Ten out of twenty kernels have $\text{OC}_{\text{T-E}}$ greater than TH = 0.1, resulting in offloading to PIM. In nine kernels among them, PIM shows higher IPC than Host.

TABLE 8. The relationship between proposed PIM-to-Host IPC and potential based offloading conditions.

Benchmark	$\text{OC}_{\text{T-E}}$	Host-to-PIM IPC ratio
Cfd_K2	1.07	3.08
Lbm_K1	0.90	0.75
Sad_K3	0.70	2.44
Sad_K2	0.67	2.98
Cfd_K1	0.54	3.01
Srad_K1	0.37	2.42
Spmv_K1	0.21	1.07
particle_K3	0.16	1.00
Stencil_K1	0.12	1.15
Hw_K1	0.11	1.40
BP_K1	0.09	0.95
Histo_K5	0.05	0.65
Histo_K1	0.05	0.63
particle_K1	0.04	0.97
Bfs_K1	0.03	0.39
BP_K1	0.02	0.42
Sgemm_K1	0.01	0.40
Sad_K1	0.008	0.40
Leukocyte_K1	0.005	0.38

Figures 9 show the results when Wen's twenty kernels are run on $\text{PIM}_{\text{PROPOSED}}$ and Wen's PIM. The performance is normalized based on Wen's baseline2. Because Wen uses 22 nm technology, the power of $\text{PIM}_{\text{PROPOSED}}$ is scaled from 28 nm to 22 nm for fair comparison. Compared to Wen's PIM with 320 GB/s memory bandwidth, $\text{PIM}_{\text{PROPOSED}}$ with slightly lower memory bandwidth of 256 GB/s achieves 19% higher IPC and 0.5% lower ED2P as shown in Figure 9.

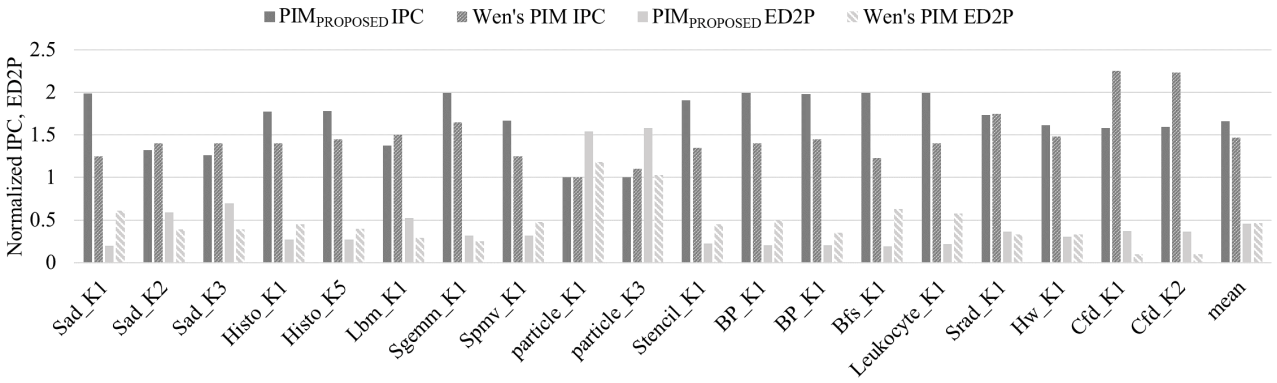

FIGURE 9. IPC and ED2P in $PIM_{PROPOSED}$ and Wen's PIM.

VIII. CONCLUSIONS

Expectations for emerging technologies PIM are very high, but they are not yet popular due to various constraints. This paper focuses on maximizing the utilization of PIM with easy-to-use task offloading conditions, minimal GPU structure changes, and metrics that can meet the power budget of PIM. The model and parameter adjustment proposed in this paper assume specific Host and PIM structures. However, the methodology itself is based on the basic data movement and power consumption of the GPU, allowing the proposed schemes to be applied to other systems with less effort. In the proposed PIM-enabled system, 11 tasks satisfying offloading conditions among 17 tasks were performed in PIM, resulting in a 20.5% increase in IPC performance and a 67.2% reduction in ED2P compared to a Host-only system. Therefore, this paper shows that the proposed offloading conditions designed by selecting intuitive and critical minimum indicators are close to the optimal performance from the previous works through intensive experiments.

REFERENCES

[1] A. Pattnaik, X. Tang, A. Jog, O. Kayiran, A. K. Mishra, M. T. Kandemir, O. Mutlu, and C. R. Das, "Scheduling techniques for GPU architectures with processing-in-memory capabilities," in *Proc. PACT*, Haifa, Israel, 2016, pp. 31–44.

[2] W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon, "Demystifying 3D ICs: The pros and cons of going vertical," *IEEE Design Test Comput.*, vol. 22, no. 6, pp. 498–510, Jun. 2005.

[3] L. Nai and H. Kim, "Instruction offloading with HMC 2.0 standard: A case study for graph traversals," in *Proc. MEMSYS*, Washington, DC, USA, 2015, pp. 258–261.

[4] R. Hadidi, L. Nai, H. Kim, and H. Kim, "CAIRO: A compiler-assisted technique for enabling instruction-level offloading of Processing-In-Memory," *ACM Trans. Archit. Code Optim.*, vol. 14, no. 4, pp. 1–25, Dec. 2017.

[5] D. Zhang, N. Jayasena, A. Lyashevsky, J. L. Greathouse, L. Xu, and M. Ignatowski, "TOP-PIM: Throughput-oriented programmable processing in memory," in *Proc. HPDC*, Vancouver, BC, Canada, 2014, pp. 23–27.

[6] Y. Wang, W. Chen, J. Yang, and T. Li, "Exploiting parallelism for CNN applications on 3D stacked Processing-In-Memory architecture," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 3, pp. 589–600, Mar. 2019.

[7] J. Ahn, S. Yoo, O. Mutlu, and K. Choi, "PIM-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture," in *Proc. ISCA*, Portland, OR, USA, 2015, pp. 336–348.

[8] Y. Wang, W. Chen, J. Yang, and T. Li, "Towards memory-efficient allocation of CNNs on Processing-in-Memory architecture," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1428–1441, Jun. 2018.

[9] Y. Tang, Y. Wang, H. Li, and X. Li, "ApproxPIM: Exploiting realistic 3D-stacked DRAM for energy-efficient processing in-memory," in *Proc. ASP-DAC*, Chiba, Japan, Jan. 2017, pp. 396–401.

[10] S. Gupta, M. Imani, H. Kaur, and T. S. Rosing, "NNPIM: A processing in-memory architecture for neural network acceleration," *IEEE Trans. Comput.*, vol. 68, no. 9, pp. 1325–1337, Sep. 2019.

[11] S. Angizi, J. Sun, W. Zhang, and D. Fan, "GraphS: A graph processing accelerator leveraging SOT-MRAM," in *Proc. IEEE DATE*, Florence, Italy, Mar. 2019, pp. 378–383.

[12] J. Choi, B. Kim, J.-Y. Jeon, H.-J. Lee, E. Lim, and C. E. Rhee, "POSTER: GPU based near data processing for image processing with pattern aware data allocation and prefetching," in *Proc. PACT*, Seattle, WA, USA, Sep. 2019, pp. 469–470.

[13] C. Xie, S. L. Song, J. Wang, W. Zhang, and X. Fu, "Processing-in-memory enabled graphics processors for 3D rendering," in *Proc. HPCA*, Austin, TX, USA, Feb. 2017, pp. 637–648.

[14] T. Vincon, A. Koch, and I. Petrov, "Moving processing to data: On the influence of processing in memory on data management," 2019, *arXiv:1905.04767*. [Online]. Available: http://arxiv.org/abs/1905.04767

[15] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "Processing data where it makes sense: Enabling in-memory computation," 2019, *arXiv:1903.03988*. [Online]. Available: http://arxiv.org/abs/1903.03988

[16] Y. Eckert, N. Jayasena, and G. H. Loh, "Thermal feasibility of die-stacked processing in memory," in *Proc. 2nd Workshop Near-Data Process.*, 2014, pp. 1–5.

[17] L. Nai, R. Hadidi, H. Xiao, H. Kim, J. Sim, and H. Kim, "Thermal-aware processing-in-memory instruction offloading," *J. Parallel Distrib. Comput.*, vol. 130, pp. 193–207, Aug. 2019.

[18] W. Wen, J. Yang, and Y. Zhang, "Optimizing power efficiency for 3D stacked GPU-in-memory architecture," *Microprocess. Microsyst.*, vol. 49, pp. 44–53, Mar. 2017.

[19] M. Scrbak, J. L. Greathouse, N. Jayasena, and K. Kavi, "DVFS space exploration in power constrained processing-in-memory systems," in *Proc. ARCS*, vol. 10172, 2017, pp. 221–233.

[20] D. P. Zhang, N. Jayasena, A. Lyashevsky, J. Greathouse, M. Meswani, M. Nutter, and M. Ignatowski, "A new perspective on processing-in-memory architecture design," in *Proc. Memory Syst. Perform. Correctness*, New York, NY, USA, 2013, pp. 1–3.

[21] H. Kim, H. Kim, S. Yalamanchili, and A. F. Rodrigues, "Understanding energy aspects of processing-near-memory for HPC workloads," in *Proc. 2015 Int. Symp. MEMSYS*, Oct. 2015, pp. 276–282.

[22] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, Apr. 2009, pp. 163–174.

[23] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Oct. 2009, pp. 44–54, doi: 10.1109/IISWC.2009.5306797.

[24] S. Grauer-Gray, L. Xu, R. Searles, S. Ayalasomayajula, and J. Cavazos, "Auto-tuning a high-level language targeted to GPU codes," in *Proc. Innov. Parallel Comput. (InPar)*, May 2012, pp. 1–10.
[25] Nvidia Corporation. *CUDA Samples, Version 4.2*. Accessed: Apr. 2, 2020. [Online]. Available: https://docs.nvidia.com/cuda/cuda-samples/index.html#samples-reference
[26] J. Kim and Y. Kim, "HBM: Memory solution for bandwidth-hungry processors," in *Proc. HCS*, Cupertino, CA, USA, Aug. 2014, pp. 1–24.
[27] *GEM5 Simulator*. [Online]. Available: http://www.gem5.org/
[28] J. Lucas, S. Lal, M. Andersch, M. Alvarez-Mesa, and B. Juurlink, "How a single chip causes massive power bills GPUSimPow: A GPGPU power simulator," in *Proc. IEEE ISPASS*, Apr. 2013, pp. 97–106.
[29] S. Song, M. Lee, J. Kim, W. Seo, Y. Cho, and S. Ryu, "Energy-efficient scheduling for memory-intensive GPGPU workloads," in *Proc. IEEE DATE*, Mar. 2014, pp. 1–6.
[30] J. Hongshin. *HBM (High Bandwidth Memory) for 2.5D*. Accessed: Apr. 2, 2020. [Online]. Available: http://www.semicontaiwan.org/zh/sites/semicontaiwan.org/files/data15/docs/4_5_semicon_taiwan_2015_ppt_template_sk_hynix_hbm_r5.pdf
[31] R. Balasubramonian, A. B. Kahng, N. Muralimanohar, A. Shafiee, and V. Srinivas, "CACTI 7: New tools for interconnect exploration in innovative off-chip memories," *ACM Trans. Archit. Code Optim.*, vol. 14, no. 2, pp. 1–25, Jun. 2017.
[32] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchitecture (Micro)*, Dec. 2009, pp. 469–480.
[33] C. Li, S. L. Song, H. Dai, A. Sidelnik, S. K. S. Hari, and H. Zhou, "Locality-driven dynamic GPU cache bypassing," in *Proc. ACM ICS*, New York, NY, USA, 2015, pp. 67–77.
[34] S. Dublish, V. Nagarajan, and N. Topham, "Evaluating and mitigating bandwidth bottlenecks across the memory hierarchy in GPUs," in *Proc. IEEE ISPASS*, Santa Rosa, CA, USA, Apr. 2017, pp. 239–248.
[35] G. Koo, H. Jeon, and M. Annavaram, "Revealing critical loads and hidden data locality in GPGPU applications," in *Proc. IEEE IISWC*, Atlanta, GA, USA, Oct. 2015, pp. 120–129.

BYOUNG-HAK KIM received the B.S. degree in information and communication engineering from Inha University, Incheon, South Korea, in 2018, where he is currently pursuing the master's degree with the Department of Information and Communication Engineering. His research interests include near data processing and VLSI design.

CHAE EUN RHEE received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2000, 2002, and 2011, respectively. From 2002 to 2005, she was with the Digital TV Development Group, Samsung Electronics Company Ltd., Suwon, South Korea, as an Engineer, where she was involved in bus architecture and MPEG decoder development. In 2013, she joined the Department of Information and Communication Engineering, Inha University, South Korea, where she is currently working as an Associate Professor. Her research interests include the algorithm and architecture design of video coding for HEVC and H.264/AVC, configurable video coding for real time systems and the next generation virtual reality systems.