electronics

MDPI

Article

Power-Time Exploration Tools for NMP-Enabled Systems

Chae Eun Rhee [1,*], Seung-Won Park [1], Jungwoo Choi [2], Hyunmin Jung [2] and Hyuk-Jae Lee [2]

[1] Department of Information and Communication Engineering, Inha University, Incheon 22212, Korea; koporia@gmail.com

[2] Department of Electrical Engineering, Seoul National University, Seoul 08826, Korea; choijw8525@capp.snu.ac.kr (J.C.); jhm6944@capp.snu.ac.kr (H.J.); hjlee@capp.snu.ac.kr (H.-J.L.)

* Correspondence: chae.rhee@inha.ac.kr; Tel.: +82-32-860-7429

Received: 15 August 2019; Accepted: 26 September 2019; Published: 28 September 2019

Abstract: Recently, dramatic improvements in memory performance have been highly required for data demanding application services such as deep learning, big data, and immersive videos. To this end, the throughput-oriented memory such as high bandwidth memory (HBM) and hybrid memory cube (HMC) has been introduced to provide a high bandwidth. For its effective use, various research efforts have been conducted. Among them, the near-memory-processing (NMP) is a concept that utilizes bandwidth and power consumption by placing computation logic near the memory. In the NMP-enabled system, a processor hierarchy consisting of hosts and NMPs is formed based on the distance from the main memory. In this paper, an evaluation tool is proposed to obtain the optimal design decision considering the power-time trade-off in the processor hierarchy. Every time the operating condition and constraints change, the decision of task-level offloading is dynamically made. For the realistic NMP-enabled system environment, the relationship among HBM, host, and NMP should be carefully considered. Hosts and NMPs are almost hidden from each other and the communications between them are extremely limited. In the simulation results, popular benchmarks and a machine learning application are used to demonstrate power-time trade-offs depending on applications and system conditions.

Keywords: high bandwidth memory; power-time-based design decision; near-memory-processing; task offloading

1. Introduction

For decades, efforts have been conducted to increase both the processor speed and memory size. Consequently, the memory bottleneck problem has become increasingly serious and is a critical issue to overcome urgently to improve overall system performance. Recently, data-intensive applications such as deep learning, big data, and immersive video have attracted attention, and a significant improvement in memory performance is in high demand. Hence, a through-silicon via (TSV)-based stacked DRAM memory such as the high bandwidth memory (HBM) [1] or hybrid memory cube (HMC) [2] has been introduced to provide a high bandwidth with a wide I/O. This next-generation memory has a structure in which multiple layers of the DRAM die are stacked on a base logic layer, and interlayer communication is achieved through high-speed TSV technology. Unlike the conventional memory, it provides a high bandwidth with low power consumption.

Various research efforts have been conducted for the effective use of the HBM and HMC. Among them, the concept of near-memory-processing (NMP) changes the traditional relationship between a processor and memory for data-intensive applications. As shown in Figure 1a, the traditional processor-centered approach has a deep memory hierarchy with several levels of cache. The closer

the distance to the processor, the smaller and the more expensive is the memory used for fast access. This hierarchy utilizes the data locality of the applications. The distance between the data location and the processor is determined by how soon the data will be used. In such a structure, it is essential to provide the necessary data quickly. In Figure 1b, the NMP exploits the bandwidth and power consumption by placing the computation logic near the memory. Hosts outside the memory and NMP inside the memory demonstrate a processor hierarchy depending on the distance from the memory. Usually, lightweight processors are considered for the NMP, whereas powerful processors are suitable for the host. When a large amount of data requires a simple calculation or when one data point does not require repeated computations, it is appropriate to use NMP. On the contrary, when a highly complex computation is necessary or when a sophisticated cache coherence protocol is essential among processors, it is reasonable to handle it at the host, at the sacrifice of an additional overhead to pass through the memory I/ O interface. Over the decades, a number of studies regarding optimal cache size, type, and its replacement policy for a processor-centered approach [3–6] have been reported. Furthermore, many design exploration and evaluation methods have been attempted to optimize the performance of multi-core environments [7,8]. Therefore, it is opportune to study various tools and schemes for system optimization and performance evaluation considering the processor hierarchy in the memory-centered approach.

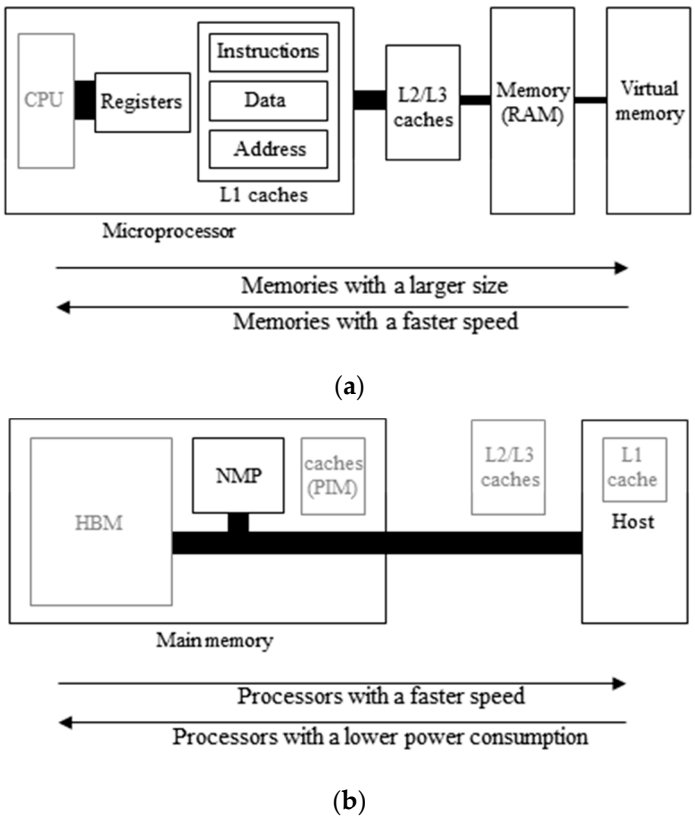

Figure 1. Processor-centered and memory-centered approaches to overcome the memory bottleneck, (**a**) deep memory hierarchy, and (**b**) processor hierarchy.

2. Previous Works and Problem Statements

With the commercialization of a three-dimensional (3D) stacked memory, research on the NMP is at a new turning point. Recently, the accurate modeling of the NMP performance is the primary research topic because no products or simulators are available that can implement NMP. Some studies have been reported that estimated and modeled the expected execution time and energy efficiency in an NMP-enabled system using the actual results obtained from the existing processors [9,10].

They conducted the performance comparison between a host and NMP on the assumption that NMP has a much higher memory bandwidth than the host [9–12]. For specific benchmarks, an NMP showed competitiveness by proving the possibility of performance enhancement due to its high bandwidth advantage [9,10,13]. Various computation-bound or memory-bound benchmarks were tested in hosts and NMPs. Subsequently, the performance in terms of execution time and energy efficiency was analyzed in relation to the benchmark characteristics. Depending on the nature of systems or benchmarks, running the host may have an absolute performance advantage over the NMP processor and vice versa. However, it is typical to obtain the optimal solution through NMP offloading in the host-NMP heterogeneous system [14–16]. Operation-level NMP offloading was tested using the MapReduced benchmark, where the trends of power consumption and execution time change are analyzed to obtain the optimal NMP offloading point [14,15]. Similarly, NMP offloading is tried in the operation level using the graph processing benchmark [16]. Given the target benchmark, the performance of an NMP-enabled system is evaluated with various configurations. By changing the memory bandwidth, operating frequency, number of processors, and cache size, the NMP structure that is suitable for the target benchmark is evaluated [10,17,18]. The energy-delay product indicator is well known and used to compare various NMP structures in two areas of interest: execution time and energy efficiency [10]. Furthermore, attempts to obtain the point at which execution time and data parallelism are maximized are conducted by changing the number and configuration of NMP [18]. In References [19–21], the possible benefits from enabling in-memory computations are well analyzed using various architecture and applications.

Investigations on the effective use of HBM are still in its infancy. The problems to be complemented are summarized as follows. First, the combination of HBM and NMP is expected to be widely used. However, realistic constraints are not considered carefully. For example, in the HBM, it is reasonable to assume that both the host and NMP are connected to the logic die's memory controller and have the same bandwidth. However, many previous studies assume a stacked memory environment in which the NMP has an absolute bandwidth advantage over the host. Next, the decision on which core to use is typically made in the application level. In other cases, an operation-level allocation decision is made, by assuming the NMP is on the very small logic die. Decision units are too coarse or too fine to obtain optimal offloading points. In addition, no criterion based on power-time tradeoff exists for the design decision and evaluation, even though the primary characteristic that NMP differentiates from the host is low power consumption. Scalability near the memory processing [22] was tried but no analytical tool was provided.

This paper focuses on the advantage of the power reduction due to the processor hierarchy in the NMP-enabled system. The primary contributions are as follows. First, the proposed tool makes it possible to test various design decisions in the early stage for the system with a processor hierarchy without the actual experiment. In this case, the power consumption and execution time are the main constrains to consider. Second, the design close to the optimal level for the whole application execution was found by searching the best operating condition at the task-level.

3. Proposed Power-Time Exploration

The HBM-based NMP-enabled system assumed in this paper contain different constraints unlike the conventional multi-core systems. First, NMP and the host are independent processors. However, the host has absolute priority for memory access. For the host, the NMP is not explicitly present and, therefore, does not compete for memory accesses, such as arbitration. After the memory controller in the host side sends the memory request, the deterministic response time should not be changed because of the NMP inside the memory. Next, the communication between the host and NMP is highly limited. They are not connected through a bus. Thus, a cache coherence protocol that is typically used in multi-core systems cannot be adopted. The idea that an NMP inside the memory uses an additional pin to send an interrupting signal to the host is also not welcome due to the hardware overhead. It is, therefore, reasonable to minimize the communication overhead between the host and

NMP, and between one NMP and another by isolating the memory access time and space. Furthermore, it is necessary to reduce the synchronization overhead by using a large processing unit such as a task, rather than a small unit such as an operation.

3.1. Power-Time Optimization

In this paper, the Lagrangian core-selection technique is adopted. The popularity of this approach in various research fields is due to its effectiveness and simplicity. In this section, the Lagrangian optimization techniques and their application to task-level NMP offloading is briefly reviewed. Let A be a specific application, and C be the assigned cores for A. The purpose of core-task allocation is to obtain a C that minimizes the whole execution time T of A. However, for the NMP-enabled system where the heat problem is critical, the total power consumption constraint P_c should be satisfied. The problem to solve can be formulated as Equation (1).

$$\begin{aligned} &\text{minimize } T(C), \\ &\text{subject to } P(C) \leq P_c \end{aligned} \tag{1}$$

T(C) is the execution time when A is executed in the core set C, whereas *P(C)* is the power consumed at that time. In practice, rather than solving the constrained problem in Equation (1), it can be reformulated with unconstrained minimization, which is shown in Equation (2). In this case, λ is the Lagrange multiplier. The solution C^* is optimal in the sense that the total execution time *T(C*)* has the minimum value with C^* among all combinations of core allocation options, which satisfies power consumption less than or equal to P_c. In this scenario, it is assumed that a power constraint P_c corresponds to λ [23].

$$C^* = \text{argmin} T(C) + \lambda \times P(C) (\lambda \geq 0) \tag{2}$$

If application A is partitioned into a number of tasks A_i ($i = 1, \ldots, n$), the associated core allocation decisions are independent of each other. An additive time measure is used assuming a serialized execution manner. The minimization problem in Equation (2) can be written as Equation (3).

$$\text{Minimize} \sum_{i=1}^{n} T_i(C_i) + \lambda \times P_c(C_i) \tag{3}$$

In this case, $T_i(C_i)$ and $P_i(C_i)$ are the time and power consumed when task A_i is allocated to core C_i, respectively. The optimum solution of Equation (3) is obtained by selecting the appropriate core C_i for each task A_i. Herein, the problem can be simplified as shown in Equation (3), by assuming task-level serial execution and considering the distinctive relation among NMP, host, and memory mentioned in the beginning of Section 3.

3.2. Power-Time Cost Using Easy-to-Use Lambda for Design Decision

In the NMP-enabled system, there are various combinations of core allocations that determine whether to execute the tasks on the host or NMP. For example, when running N tasks to a system consisting of one host and one NMP, a combination of 2^N exists. Among them, the best decision should be made considering power and time tradeoff. As N increases, the available core allocation combination increases rapidly. Searching for all cases is time consuming. To narrow the search range for the core allocation, the Lagrangian optimization scheme is adopted. Equation (3) is represented by the cost function as Equation (4). The best offloading decision is made by choosing the host or NMP to have a low cost for each task. The searching complexity is reduced from 2^N to $2N$.

$$\text{Cost} = \sum_{i=1}^{n} T_i(C_i) + \lambda \times P_i(C_i) \tag{4}$$

To solve the minimization problem shown in Equation (4), it is important to determine the λ value. Given the cost function, if T is differentiated with respect to P, the best offloading point is determined by obtaining the stationary point. In other words, the most convex point needs to be found in the power-time curve shown in Figure 2. However, in the simulation-based design exploration, a power-time curve can be obtained after testing all the numerous offloading options. As the numbers of core and task increase, the computational complexity increases sharply. Note that the best offloading point is determined by the relative difference in the performance of the NMP and the host. It is observed that, for a given application, the shape and slope of the power-time curve are primarily determined by the performance ratio of the host and the NMP. The lower the performance of the NMP is, the higher the slope of the curve will be. In contrast, the higher the performance of the host is, the lower the slope of the curve will be. Fortunately, in the NMP-enabled system, both end points of the curve are easily obtained. One end point represents an application running in the host-only system, whereas the other represents running in the NMP-only system. Therefore, herein, the λ value is determined based on the performance ratio of the given host and NMP as Equation (5). This approach also satisfies the qualifying condition for λ to solve the unconstrained problem [23]. The P_c constraint corresponds to λ. The value of λ increases as the power constraint P_c is decreased, whereas λ decreases when the high-performance processors are used due to the sufficient P_c.

$$\lambda = \frac{Time_{PIM\_only} - Time_{host\_only}}{Power_{host\_only} - Power_{PIM\_only}} \tag{5}$$

Figure 2 shows an example of assigning four tasks to the host and the NMP. The horizontal axis represents the power consumption, whereas the vertical axis represents the execution time. Black dots represent the power and time results of 2^4 choices. For the NMP in the base layer of the HBM, it is realistic to use a processor with low performance and low power when compared to the host. Therefore, the offloading of some tasks to the NMP increases the total execution time of the application, whereas the power consumption is decreased. A gray power-time curve consisting of dots on the lower left represents the power and time tradeoff as tasks are offloaded to the NMP. The most convex point in the curve will be the best NMP offloading choice with the least increase in the execution time versus power reduction. In this paper, the best offloading point is determined by $2 \times 4 = 8$ runs. First, the value of λ is set according to the performance ratio of NMP and host using Equation (5). The determined λ value is indicated by a dashed line in Figure 2. Then, two costs of each task executed on the host and the NMP are compared to decide whether or not to offload.

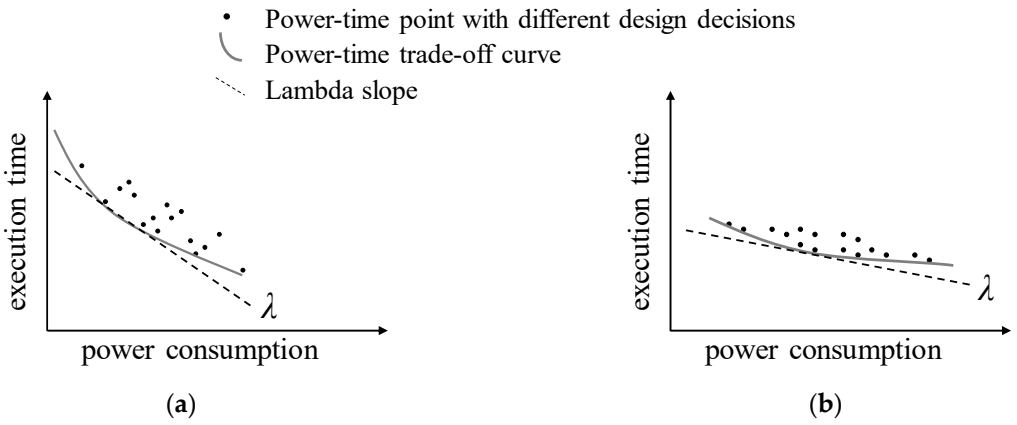

Figure 2. Power-Time curve (**a**) when a power constraint is low and (**b**) when a power constraint is high.

It is very common to change the operating condition of the systems due to the energy status or user requirements. After measuring the power and time values for each task in the initial system condition,

the changed power and time values can be estimated for various system conditions. Based on the estimated power and time data, the decision of task-level offloading is dynamically made through the proposed evaluation tool based on the power-time cost. Given the same application, Figure 2a shows the case where the power constraint P_c is small. In this case, an NMP with a low-power configuration such as a low operating clock frequency is adopted. This indicates that the processor performance of the NMP is significantly lower than that of the host. Naturally, the value of λ is high. In the most convex point of this curve, three tasks are offloaded to the NMP. For certain reasons, the required power constraint P_c is not severe such as in the case of Figure 2b. Thus, the operating clock frequency of the host is increased. The power-time curve is estimated from the actual results of Figure 2a. The value of λ, which is also estimated becomes low. In this case, in the best offloading point, two tasks are offloaded to the NMP. Therefore, one task should move to the host.

4. Evaluation Environment

4.1. NMP-Enabled System Organization

The organization of the NMP-enabled heterogeneous system assumed herein is as follows. A high-performance host processor and a low-performance NMP processor are used. The purpose of this paper is to propose a tool to measure and search the power-time performance when the HBM exhibits a processor hierarchy. Therefore, to reduce the complexity of the simulation, one host and one NMP are used. Table 1 shows the specifications of the host and NMP used in the simulation. Both are x86 processors and use a 32 KB L1 cache. To create a difference in the computing performance between the host and NMP, the host assumes an out-of-order CPU and additionally includes 1 MB L2 cache. The NMP processor, meanwhile, assumes an in-order CPU and does not include an L2 cache. No bus-like communication channel exists between the host and the NMP, and data are exchanged only through the memory. The NMP is directly connected to the 4 GB HBM2 via TSV, whereas the host can access HBM2 through the interposer and the memory I/O interface.

Table 1. Host and NMP specification.

Components	Host	NMP
CPU core parameters	Out-of-order issue 1 core, 192 entry ROB	In-order issue 1 core
L1 cache	32 KB 2 cycle access 2 way I cache and D cache	
L2 cache	1 MB 20 cycle access 8 ways	none

As mentioned earlier, the host must have absolute priority in the memory, and its memory access response time should not be affected by the NMP. To ensure the reliability of the system with these restrictions, the host and NMP do not access the memory simultaneously. Furthermore, the host-NMP organization requires a way to avoid a cache coherence violation because they do not use the cache coherence protocol through the bus. In this study, it is assumed that the cache write-backs the data and flushes itself when the core accessing memory changes. Through the above process, the cache of the newly accessing core is always empty, and the data of the previous core is recorded in the main memory. Therefore, a coherence problem does not occur. The cache flush may cause a performance degradation due to an increase in the cache miss. However, the overhead die to the cache flush will not be large because the shared data between tasks are already minimized when dividing an application into tasks.

4.2. Simulation Environment

It is time consuming to simulate various NMP offloading cases of all tasks to obtain the power and time. Therefore, for simplicity, each task is performed in the host-only and NMP-only environments. Subsequently, the power and time in NMP offloading are calculated by combining the results obtained in task units. To compare the optimal offloading point for various system conditions, the operating clock frequency is changed. For the host processor, the simulation is performed for four clock frequencies of 1, 2, 3, and 4 GHz, whereas, for the NMP processor, four clock frequencies of 200, 400, 600, and 800 MHz are used.

The execution time of each task is measured in the Gem5 full system mode [24], whereas the power consumptions of the processors and caches are calculated using McPAT [25]. The supply voltage changes according to the operating clock frequency [26]. The 32-nm logic technology is assumed. The DRAM power consumption is modeled as a DRAM layer and logic layer, separately. When the host accesses the HBM, the energy per bit is known to be 6 to 7 pJ/bit [27]. Conservatively, by assuming that the energy consumed by the DRAM layer and TSV is similar to HMC, it is estimated that 3.7 pJ/bit [28] and 2.3 pJ/bit are consumed in the logic layers, respectively. The NMP accesses the HBM directly without using an I/O interface. Thus, only the energy required for the DRAM layer access is considered for the power consumption calculation. In the case of DRAM static power, no difference is observed between the NMP and host. The power consumption is calculated using Equation (6) through the DRAM energy per bit, i.e., pJ/bit = mw/Gbps.

$$\text{TotalPower} = \text{CPU}_{\text{dynamic\_power}} + \text{CPU}_{\text{static\_power}} + \text{DRAM}_{\text{power\_per\_bit}} \times \text{Bandwidth} \quad (6)$$

Three applications are used for the simulation. PARSEC [29] and SPLASH-2 [30] are widely used as benchmarks in multi-core environments, and VGG-F [31] is a CNN algorithm of recent interest. To construct an application consisting of tasks with different characteristics, each benchmark of PARSEC is used as a task. PARSEC is composed of various benchmarks with characteristics that are computation-intensive or data-intensive. Different benefits can be obtained from the host or NMP according to the benchmark, which is suitable for observing the performance change due to NMP offloading. In this simulation, nine benchmarks of *bodytrack*, *canneal*, *dedup*, *fluidanimate*, *freqmine*, *streamcluster*, *swaptions*, *vips*, and *x264* are used. SPLASH-2 is primarily used in distributed shared memory machines and is dominated by high-performance computation and graphics-intensive benchmarks. As a second application, a computation-bound application is composed of 10 benchmarks known as *barnes*, *cholesky*, *fft*, *fmm*, *lu_cb*, *ocean*, *radiosity*, *radix*, *volrend*, and *water_nsquared* from SPLASH-2. For the third application, a machine-learning application known as data-demanding is chosen. VGG-F is a type of CNN developed by the VGG group. In this study, the VGG-F is divided into eleven tasks based on the layer. The 11 tasks correspond to five convolution layers, three fully connected layers, a relu layer with softmax, a local response normalization, and a max pooling layer, respectively.

5. Results and Discussion

The proposed cost-based search and the full search schemes are compared for PARSEC, SPLASH-2, and VGG-F. Simulations are performed on four system configurations to compare the accuracy in various power constraints. In each configuration, the NMP and the host have the following operating clock frequencies of (200 MHz, 1 GHz), (400 MHz, 1 GHz), (400 MHz, 3 GHz), and (600 MHz, 4 GHz). In Figure 3, the horizontal axis represents the power consumption, whereas the vertical axis represents the execution time. The best offloading points are obtained in four configurations and the power-time values at that time are shown. Although the full search curve is slightly lower on the left side than the proposed curve, the two curves tend to be similar. Recall that the proposed and full search schemes need 2N and 2^N computations, respectively, given N tasks. PARSEC, SPLASH-2, and VGG-F consists of nine, 10, and 11 tasks, respectively. In the full search scheme, 512, 1024, and 2048 computations are

necessary for three applications, respectively. When the proposed scheme is used, the computational complexity is reduced by about 98%.

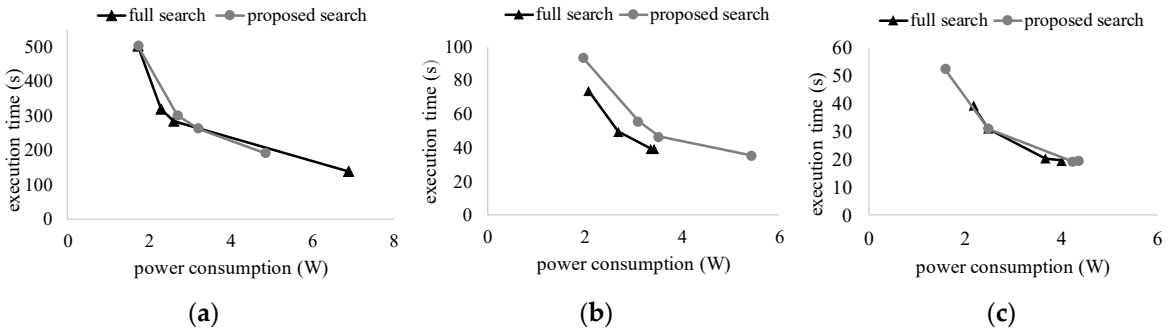

Figure 3. Comparison of NMP offloading decision between the proposed eight-search and the full 16-search (**a**) PARSEC, (**b**) SPLASH-2, and (**c**) VGG-F.

In Table 2, 16 system configurations are considered. The host can operate with the clock frequencies of 1, 2, 3, and 4 GHz, whereas the NMP can operate with the clock frequencies of 200, 400, 600, and 800 MHz. For three applications, the λ value, the number of NMP offloading tasks, and its power and time values are shown. Since the performance of the host or NMP processor is higher, the power-time curve moves to the lower right direction. Thus, the λ value also decreases. At the small λ value, it can be predicted that the task's NMP offloading is less preferred. For example, when the NMP operates at 200 MHz, the λ decreases as the host processor performance increases, and the number of NMP offloading tasks decreases. In most cases, the optimal power-time moves to the lower right. However, when the performance of the host processor is low and the performance of the NMP processor is high, i.e., the performance difference between the two processors is not extremely large, the number of NMP offloading tasks and the optimal power-time points are not well predicted. For example, in a PARSEC application, when the host performance is 1 GHz, the λ value decreases as the operating clock frequency of the NMP increases. However, the number of NMP offloading tasks increases, and the optimal power-time point decision is also inconsistent. This is because the difference between the hosts or the NMP selection is not large when it is more important to reduce the power consumption than the execution time.

Table 3 shows the results of dynamically determined offloading when the system condition is changed. The presented data are similar to Table 2. However, the power-time data of all tasks required to decide the best offloading point is not obtained by simulation but is estimated. In this simulation, the initial system configuration has the host with 2 GHz and the NMP with 400 MHz. Thus, the results of shaded cells in Table 3 are from the actual simulation data and it is exactly the same with the values in Table 2. To decide the offloading point in other system configurations, the required power-time data are scaled from the initial system condition without a running simulation. For example, suppose that the operating clock frequencies of the host and NMP are increased to 4 GHz and 600 MHz, respectively. The execution time is proportional to the operating clock frequency. Thus, the execution time of tasks on the host and the NMP are obtained by multiplying the initial data by 2 and 1.5 times, respectively. In the case of power data in the NMP and host, it is calculated as 1.386 times and 1.163 times as increasing by 200 MHz and 1 GHz, respectively. The scaling factor is obtained empirically. When comparing with the actually obtained data in Table 2, the errors of estimated power and time data at the best offloading point are 19% and 12%, respectively. However, the difference in the number of offloaded tasks is very small, which is, on average, 0.3 task.

Table 2. The best offloading point from the actual data and its power-time performance.

		200 MHz				400 MHz				600 MHz				800 MHz			
		λ	Tasks	Power	Time	λ	Tasks	Power	Time	λ	Tasks	Power	Time	λ	Tasks	Power	Time
PARSEC	1 GHz	32.40	4	1.97	93.32	15.69	4	3.10	55.72	10.38	6	1.83	63.42	8.13	6	2.07	56.20
	2 GHz	25.99	3	2.45	60.27	12.99	4	2.79	48.76	8.85	4	3.51	37.95	7.11	4	4.15	32.27
	3 GHz	16.79	3	3.04	58.85	8.46	4	3.53	46.61	5.80	4	4.57	35.97	4.68	4	5.34	29.97
	4 GHz	13.21	3	3.57	58.21	6.67	3	6.06	33.08	4.58	4	5.43	35.44	3.70	4	6.07	28.80
SPLASH-2	1 GHz	20.08	4	1.59	52.34	10.79	4	2.48	30.98	7.23	4	3.16	23.97	5.44	5	2.94	24.39
	2 GHz	15.98	3	2.02	35.41	8.75	4	2.28	28.91	5.97	4	3.04	21.14	4.58	4	3.70	17.23
	3 GHz	10.08	3	2.51	34.24	5.55	3	4.23	19.29	3.80	4	3.87	20.05	2.92	4	4.77	16.15
	4 GHz	7.76	3	2.79	33.65	4.28	3	4.79	18.70	2.94	4	4.36	19.50	2.26	4	5.41	15.60
VGG-F	1 GHz	157.67	8	1.73	502.40	75.12	8	2.69	301.22	47.68	8	3.41	234.28	34.33	8	3.95	201.01
	2 GHz	129.16	7	2.88	275.42	63.50	8	2.49	272.95	41.66	8	3.23	206.06	31.04	8	3.83	172.79
	3 GHz	83.83	7	3.76	264.18	41.58	8	3.20	263.58	27.48	8	4.23	196.69	20.63	8	5.06	163.42
	4 GHz	66.01	7	4.32	258.10	32.90	8	3.65	258.52	21.85	8	4.86	191.63	16.47	8	5.85	158.35

Table 3. The best offloading point from the estimated data and its power-time performance.

		200 MHz				400 MHz				600 MHz				800 MHz			
		λ	Tasks	Power	Time	λ	Tasks	Power	Time	λ	Tasks	Power	Time	λ	Tasks	Power	Time
PARSEC	1 GHz	36.21	4	2.05	97.51	16.65	6	1.52	86.66	10.13	4	3.12	45.72	6.86	4	3.71	38.09
	2 GHz	26.97	3	2.40	60.49	12.99	4	2.79	48.76	8.33	4	3.69	36.05	6.00	6	2.10	43.33
	3 GHz	19.59	3	2.39	56.72	9.56	3	4.03	32.13	6.22	4	3.81	32.50	4.55	4	4.68	26.15
	4 GHz	14.15	3	2.56	54.84	6.95	3	4.44	30.24	4.55	3	6.00	22.05	3.35	4	5.20	24.38
SPLASH-2	1 GHz	24.39	4	1.68	57.82	11.56	4	2.56	32.33	7.28	5	2.60	29.56	5.15	5	3.13	24.13
	2 GHz	17.92	3	1.99	35.65	8.75	4	2.28	28.91	5.69	4	2.89	20.13	4.17	4	3.55	16.17
	3 GHz	12.96	3	1.95	33.89	6.38	3	3.34	18.70	4.19	4	3.09	19.27	3.09	4	3.62	14.74
	4 GHz	9.34	3	2.07	33.01	4.62	3	3.64	17.82	3.04	3	4.99	12.76	2.26	4	4.21	14.46
VGG-F	1 GHz	177.08	8	1.83	545.91	79.18	8	2.89	327.21	46.51	9	2.56	304.15	30.15	9	3.08	249.71
	2 GHz	133.49	7	2.76	286.90	63.50	8	2.49	272.95	40.19	8	3.31	200.06	28.53	8	4.00	163.61
	3 GHz	97.29	7	2.78	265.13	47.11	7	4.60	154.34	30.40	8	3.38	181.97	22.04	8	4.18	145.52
	4 GHz	36.21	4	2.05	97.51	16.65	6	1.52	86.66	10.13	4	3.12	45.72	6.86	4	3.71	38.09

6. Conclusions

In this study, task offloading from host to NMP was attempted in an NMP-enabled system, and a power-time-based evaluation tool for an optimal design decision was proposed. In a high-performance processor, heat due to power consumption is a serious problem that can degrade the operation speed unexpectedly. The advantages of the host and NMP are high performance and low power consumption, respectively, and this is distinguished from conventional multi-core heterogeneous systems. Therefore, the search for the optimal point considering the power-time tradeoff is very meaningful in NMP-enabled systems. The power-time performance is also an effective metric for comparison with other systems such as the host-only or NMP-only system. In the simulation of this paper, only the operating clock frequency was used to change the system condition, and it had to be complemented subsequently. When introducing various processor types such as a GPU or hardware accelerator, or when the cache size or its structure changes, the power-time curve will vary significantly. Hence, it is expected that the scope of design exploration will be further expanded. When the proposed scheme is extended for architectural designs with more complex and deeper processor layers, only simple conditions should be considered further. The reason for this is due to the NMP having difficulty communicating with other processors besides the host because it is hard to use additional physical pins due to the situation of processing units inside the memory.

Author Contributions: Conceptualization, C.E.R. and J.C.; methodology, C.E.R.; software, S.-W.P.; validation, S.-W.P., J.C. and H.J.; formal analysis, J.C.; investigation, S.-W.P.; resources, H.J.; data curation, H.J.; writing—original draft preparation, C.E.R.; writing—review and editing, C.E.R. and J.C.; visualization, H.J.; supervision, C.E.R.; project administration, C.E.R. and H.-J.L.; funding acquisition, C.E.R. and H.-J.L.

Funding: The R&D program of MOTIE/KEIT and the NLR program of MOST/KOSEF supported this work [No. 10077609, Developing Processor-Memory-Storage Integrated Architecture for Low Power, High Performance Big Data Servers and by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2018R1A1A1A05023598)].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Standard, J. *High Bandwidth Memory (hbm) Dram*; JEDEC: Arlington, VA, USA, 2013.
2. Pawlowski, J.T. Hybrid memory cube (hmc). In Proceedings of the 2011 IEEE Hot Chips 23 Symposium (HCS), Stanford, CA, USA, 17–19 August 2011; pp. 1–24.
3. Stone, H.S.; Turek, J.; Wolf, J.L. Optimal partitioning of cache memory. *IEEE Trans. Comput.* **1992**, *41*, 1054–1068. [CrossRef]
4. Jeong, J.; Dubois, M. Cache replacement algorithms with nonuniform miss costs. *IEEE Trans. Comput.* **2006**, *55*, 353–365. [CrossRef]
5. Cai, Y.; Schmitz, M.T.; Ejlali, A.; Al-Hashimi, B.M.; Reddy, S.M. Cache size selection for performance, energy and reliability of time-constrained systems. In Proceedings of the 2006 Asia and South Pacific Design Automation Conference, Yokohama, Japan, 24–27, January 2006; pp. 923–928.
6. Al-Zoubi, H.; Milenkovic, A.; Milenkovic, M. Performance evaluation of cache replacement policies for the spec cpu2000 benchmark suite. In Proceedings of the 42nd Annual Southeast Regional Conference, Huntsville, AL, USA, 2–3 April 2004; pp. 267–272.
7. Monchiero, M.; Canal, R.; Gonzalez, A. Power/performance/thermal design-space exploration for multicore architectures. *IEEE Trans. Parallel Distrib. Syst.* **2008**, *19*, 666–681. [CrossRef]
8. Hanumaiah, V.; Vrudhula, S. Energy-efficient operation of multicore processors by dvfs, task migration, and active cooling. *IEEE Trans. Comput.* **2014**, *63*, 349–360. [CrossRef]
9. Zhang, D.; Jayasena, N.; Lyashevsky, A.; Greathouse, J.L.; Xu, L.; Ignatowski, M. Top-PIM: Throughput-oriented programmable processing in memory. In Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing, Vancouver, BC, Canada, 23–27 June 2014; pp. 85–98.
10. Scrbak, M.; Greathouse, J.L.; Jayasena, N.; Kavi, K. Dvfs space exploration in power constrained processing-in-memory systems. In Proceedings of the International Conference on Architecture of Computing Systems, Vienna, Austria, 3–6 April 2017; pp. 221–233.

11. Kang, Y.; Huang, W.; Yoo, S.-M.; Keen, D.; Ge, Z.; Lam, V.; Pattnaik, P.; Torrellas, J. Flexram: Toward an advanced intelligent memory system. In Proceedings of the 1999 IEEE International Conference on Computer Design: VLSI in Computers and Processors, Austin, TX, USA, 10–13 October 2012; pp. 5–14.
12. Draper, J.; Chame, J.; Hall, M.; Steele, C.; Barrett, T.; LaCoss, J.; Granacki, J.; Shin, J.; Chen, C.; Kang, C.W.; et al. The architecture of the diva processing-in-memory chip. In Proceedings of the 16th International Conference on Supercomputing, New York, NY, USA, 22–26 June 2002; pp. 14–25.
13. Gries, M.; Cabré, P.; Gago, J. *Performance Evaluation and Feasibility Study of Near-data Processing on DRAM Modules (DIMM-NDP) for Scientific Applications*; Huawei Technologies Duesseldorf GmbH, Munich Research Center (MRC): München, Germany, 2019.
14. 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 Proceedings of the 2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Monterey, CA, USA, 23–25 March 2014; pp. 190–200.
15. Islam, M.; Scrbak, M.; Kavi, K.M.; Ignatowski, M.; Jayasena, N. Improving node-level mapreduce performance using processing-in-memory technologies. In Proceedings of the European Conference on Parallel Processing, Porto, Portugal, 25–26 August 2014; pp. 425–437.
16. Ahn, J.; Hong, S.; Yoo, S.; Mutlu, O.; Choi, K. A scalable processing-in-memory accelerator for parallel graph processing. *ACM SIGARCH Comput. Archit. News* **2015**, *43*, 105–117. [CrossRef]
17. Scrbak, M.; Islam, M.; Kavi, K.M.; Ignatowski, M.; Jayasena, N. Processing-in-memory: Exploring the design space. In Proceedings of the International Conference on Architecture of Computing Systems, Porto, Portugal, 24–27 March 2015; pp. 43–54.
18. Xu, L.; Zhang, D.P.; Jayasena, N. Scaling deep learning on multiple in-memory processors. In Proceedings of the 3rd Workshop on Near-Data Processing, Waikiki, HI, USA, 5–9 December 2015.
19. Vincon, T.; Koch, A.; Petrov, I. Moving Processing to Data: On the Influence of Processing in Memory on Data Management. *arXiv* **2019**, arXiv:190504767. Available online: https://arxiv.org/abs/1905.04767 (accessed on 15 August 2019).
20. Mutlu, O.; Ghose, S.; Gómez-Luna, J.; Ausavarungnirun, R. Processing data where it makes sense: Enabling in-memory computation. *Microprocess. Microsyst.* **2019**, *67*, 28–41. [CrossRef]
21. Awan, J.; Ohara, M.; Ayguade, E.; Ishizaki, K.; Brorsson, M.; Vlassov, V. Identifying the potential of near data processing for apache spark. In Proceedings of the International Symposium on Memory Systems, Alexandria, VA, USA, 2–5 October 2017; pp. 60–67.
22. Dysart, T.; Kogge, P.; Deneroff, M.; Bovell, E.; Briggs, P.; Brockman, J.; Jacobsen, K.; Juan, Y.; Kuntz, S.; Lethin, R.; et al. Highly scalable near memory processing with migrating threads on the emu system architecture. In Proceedings of the Sixth Workshop on Irregular Applications: Architectures and Algorithms, Salt Lake City, UT, USA, 13–18 November 2016; pp. 2–9.
23. Wiegand, T.; Schwarz, H.; Joch, A.; Kossentini, F.; Sullivan, G.J. Rate-constrained coder control and comparison of video coding standards. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 688–703. [CrossRef]
24. Binkert, N.; Beckmann, B.; Black, G.; Reinhardt, S.K.; Saidi, A.; Basu, A.; Hestness, J.; Hower, D.R.; Krishna, T.; Sardashti, S.; et al. The gem5 simulator. *ACM SIGARCH Comput. Archit. News* **2011**, *39*, 1–7. [CrossRef]
25. Li, S.; Ahn, J.H.; Strong, R.D.; Brockman, J.B.; Tullsen, D.M.; Jouppi, N.P. Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, New York, NY, USA, 12–16 December 2009; pp. 469–480.
26. Spiliopoulos, V.; Bagdia, A.; Hansson, A.; Aldworth, P.; Kaxiras, S. Introducing dvfs-management in a full-system simulator. In Proceedings of the 2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, San Francisco, CA, USA, 14–16 August 2013; pp. 535–545.
27. O'Connor, M. Highlights of the high-bandwidth memory (hbm) standard. In Proceedings of the Memory Forum Workshop, Minneapolis, MN, USA, 14 June 2014.
28. Jeddeloh, J.; Keeth, B. Hybrid memory cube new dram architecture increases density and performance. In Proceedings of the 2012 symposium on VLSI technology (VLSIT), Honolulu, HI, USA, 12–14 June 2012; pp. 87–88.

29. Bienia, C.; Kumar, S.; Singh, J.P.; Li, K. The parsec benchmark suite: Characterization and architectural implications. In Proceedings of the 17th International Conference on Parallel architectures and Compilation Techniques, Toronto, ON, Canada, 25–29 October 2008; pp. 72–81.
30. Woo, S.C.; Ohara, M.; Torrie, E.; Singh, J.P.; Gupta, A. The splash-2 programs: Characterization and methodological considerations. In Proceedings of the 22nd Annual International Symposium on Computer Architecture, S. Margherita Ligure, Italy, 22–24 June 1995; pp. 24–36.
31. Chatfield, K.; Simonyan, K.; Vedaldi, A.; Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. *arXiv* **2014**, arXiv:1405.3531. Available online: https://arxiv.org/abs/1405.3531 (accessed on 15 August 2019).