electronics

MDPI

Article

Q-Selector-Based Prefetching Method for DRAM/NVM Hybrid Main Memory System

Jeong-Geun Kim [1], Shin-Dug Kim [1] and Su-Kyung Yoon [2,3,*]

[1] Department of Computer Science, Yonsei University, Seoul 03722, Korea; junggeun@yonsei.ac.kr (J.-G.K.); sdkim@yonsei.ac.kr (S.-D.K.)
[2] Division of Computer Science and Engineering, Jeonbuk National University, Jeonju 54896, Korea
[3] Research Center for Artificial Intelligence Technology, Jeonbuk National University, Jeonju 54896, Korea
* Correspondence: sk.yoon@jbnu.ac.kr

Received: 31 October 2020; Accepted: 20 November 2020; Published: 16 December 2020

Abstract: This research is to design a Q-selector-based prefetching method for a dynamic random-access memory (DRAM)/ Phase-change memory (PCM)hybrid main memory system for memory-intensive big data applications generating irregular memory accessing streams. Specifically, the proposed method fully exploits the advantages of two-level hybrid memory systems, constructed as DRAM devices and non-volatile memory (NVM) devices. The Q-selector-based prefetching method is based on the Q-learning method, one of the reinforcement learning algorithms, which determines a near-optimal prefetcher for an application's current running phase. For this, our model analyzes real-time performance status to set the criteria for the Q-learning method. We evaluate the Q-selector-based prefetching method with workloads from data mining and data-intensive benchmark applications, PARSEC-3.0 and graphBIG. Our evaluation results show that the system achieves approximately 31% performance improvement and increases the hit ratio of the DRAM-cache layer by 46% on average compared to a PCM-only main memory system. In addition, it achieves better performance results compared to the state-of-the-art prefetcher, access map pattern matching (AMPM) prefetcher, by 14.3% reduction of execution time and 12.89% of better CPI enhancement.

Keywords: memory system; hybrid memory system; Q-learning; reinforcement learning; prefetching; phase change memory; emerging memory; non-volatile memory

1. Introduction

Recently, in the in-memory computing and big-data processing era, computing systems are suffering from a lack of main memory capacity. However, dynamic random-access memory (DRAM) technology is the dominant main memory device throughout the last few decades. However, its limitation of scaling and high cost keeps the memory space smaller than the working set size required by modern applications. To mitigate this problem, there have been several studies about how to apply new memory technologies into the system. Phase-change memory (PCM) is one of the most promising memory technologies, exhibiting high density, DRAM-comparable speed, and high cost-efficiency among the emerging memory candidates [1,2]. Despite these advantages, the PCM cannot entirely replace DRAM devices because of its longer access latency (~3 × that of DRAM). Therefore, the hybrid main memory structure formed as DRAM/PCM is one of the feasible ways to introduce emerging memory technologies for real computing systems [1–4]. Nevertheless, previous studies have focused on how to allocate data into the memory-address space [2] and tried to reduce the impact from slow PCM exploiting its characteristics [3]. Furthermore, there were inadequate attempts to apply prefetching methods for DRAM/PCM hybrid memory.

Hence, we propose a DRAM/PCM hybrid main memory system based on a selective prefetching mechanism to hide a long latency of PCM devices when the DRAM can support most of the memory requests to hit via the DRAM cache. In addition, to prevent performance degradation by many DRAM misses when the system encounters frequent changes of memory access patterns, we introduce a prefetching management method based on machine learning called the Q-selector-based prefetcher (QSBP) that selects an adequate prefetcher engine that is considered the most appropriate prefetcher to handle a current memory pattern dynamically. As a result of the system-level simulation, the QSBP reduced the total execution time by approximately 31% and increased the DRAM hit ratio by 46% on average, compared to a PCM-only main memory system that requires the same cost to configure the main memory layer. Furthermore, it shows better performance than a state-of-the-art prefetcher-based DRAM/PCM hybrid main memory system.

We make the following contributions to this work:

- We propose QSBP, a dynamic prefetcher selection method for DRAM/PCM hybrid main memory systems based on an online reinforcement learning method.
- We measure the entire performance and hit ratio of DRAM caches of various DRAM/PCM hybrid main memory models including QSBP. The result of the evaluation shows that our QSBP model is far better than the traditional hybrid main memory system without any management methods and a state-of-the-art prefetcher based model.

The remainder of this article is organized as follows. In Section 2, we introduce the background and related works about hybrid main memory systems and non-volatile memory technologies. In Section 3, we describe our proposed model, QSBP, with its motivation. In Section 4, we describe our simulation environment and target workloads. In Section 5, the results of evaluation by the simulation are provided. And finally, in Section 6, we conclude that the proposed model is effective at improving the entire performance of the hybrid main memory-based computing systems considering its construction cost.

2. Background and Related Works

2.1. Phase Change Memory

Current DRAM and NAND flash solid-state drive (SSD) cannot fill the gap between memory and storage in the meaning of cost efficiency and latency [5]. As modern data-intensive applications, such as machine learning, data mining, and in-memory processing workloads, require a large amount of working memory space that is greater than current DRAM main memory sizes. Moreover, the physical limitations of DRAM (e.g., scaling problem, refresh operation, etc.) technology act like a non-trivial obstacle to expand the size of the main memory layer.

Hence, to overcome these limitations of DRAM, many computer architects have introduced emerging memory technologies for computer architecture. These emerging memory devices are called non-volatile memory (NVM) or storage class memory (SCM) because of their nonvolatility characteristics like storage device and storage-class capacity with DRAM-comparable latency. According to W. Zhang et al [5], PCM is one of the promising NVM technologies which is made of a chalcogenide material ($Ge_2Sb_2Te_5$, called GST is the most commonly used chalcogenide material for phase-change memory devices). The GST stores data using the differences of resistance degrees by its phase [1]. PCM provides larger capacity without losing cost efficiency and is the only device being launched as a commercial on-the-shelf [3,4]. Therefore, it is believed to be the most realistic emerging memory device to replace DRAM's role in the near future. However, its slow latency (~4× that of DRAM) still prohibits replacing entire DRAM devices now. Thus, computer architects suggest using DRAM and PCM devices together, called hybrid main memory systems, to hide each device's weaknesses.

2.2. DRAM/PCM Hybrid Memory Systems

To mitigate the current limitations of PCM-only main memory systems, the concept of DRAM/PCM hybrid memory system is introduced. The most general form of organizing a hybrid memory system is using the DRAM as a cache for the slow PCM device [1,3]. The other method is to use DRAM as a part-of-memory (PoM) [2] to avoid wasting its capacity; PoM fully uses 100% of the entire hybrid main memory area (DRAM capacity + PCM capacity) without spending inclusive cache spaces. In this paper, we employ a DRAM-cache type of hybrid main memory system, because the PoM structure may cause non-trivial management overhead from maintaining a physical to device-address translation table and page-migration engine requiring periodic inter-memory operations which may degrade system performance.

As shown in Figure 1, recently Intel officially launched Optane DC persistent memory module (Optane DIMMs) for commercial markets. Optane DIMMs have two operation modes, including memory mode and AppDirect mode: the memory mode set DRAM as direct-mapped cache-like near memory for Optane (PCM) far memory, and the AppDirect mode allows users use PCM directly by persistent memory libraries (PMDK: Persistent Memory Development Kit library [6]) but not allow control the operation of internal DRAM area. AppDirect mode needs the modifying codes of applications to exploit the non-volatility of Optane devices with slow PCM memory latency, but the memory mode can run any un-modified workloads with enhanced memory access latency supported by DRAM cache.

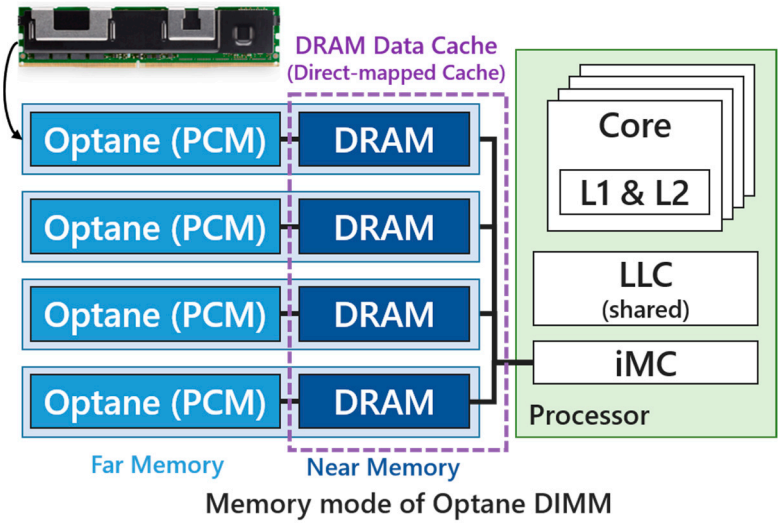

Figure 1. The overview of Intel® Optane™ dual in-line memory module (DIMM)-based memory system. Optane DIMM has two operation modes: memory mode and AppDirect mode; the memory mode is a type of hybrid memory that configures PCM as far memory and DRAM as data cache.

As shown in Table 1, some non-volatile memory technologies are considered as a next-generation memory device, used as storage systems to cache memory systems. NAND Flash memory is the most matured NVM technology, which is widely used in current storage media. Similar to the DRAM main memory system, constructing resistance random access memory (ReRAM) or spin-transfer torque magnetoresistive random access memory (STT-MRAM) as a working memory with a DRAM-based write buffer is a feasible design to hide the long latency of ReRAM and STT-MRAM write operations. To reduce the gap between read and write latency, a dirty-data-only-write operation could be adopted, and the buffer contains information of validity and clean/dirty flag. However, the read latencies of ReRAM and STT-MRAM are similar to DRAM's read latency, hence, there is no additional management method like cache-like two-level memory hierarchy or data migration between DRAM and NVM device. In addition, ReRAM and STT-MRAM show almost infinite write endurance than NAND Flash

and PCM devices; it does not need wear-leveling technologies at all. Thus, ReRAM and STT-MRAM are good emerging main memory candidates (or a part of hybrid main memory systems) by their performance characteristics, but its maturity and density (STT-MRAM has a low density compared to other NVM devices) are current difficulties in using these devices as a part of the main memory area.

Table 1. Various non-volatile memory (NVM) technologies with internal parameters [7].

	NAND Flash	PCM	ReRAM	STT-MRAM
Read Latency	15–35 us	20–60 ns	~10 ns	2–35 ns
Write Latency	200–500 us	20–150 ns	~50 ns	3–50 ns
Dynamic Energy (R/W)	Low	Medium/High	Low/High	Low/High
Write Endurance	10^4–10^5	10^8–10^9	10^8–10^{11}	10^{12}–10^{15}
Maturity	Mature	Mature (Intel Optane)	Test Chips	Test Chips

On the other hand, phase change memory (PCM) is one of the most promising technologies for the near-future memory architecture, and the commercial version of PCM devices are already launched by Intel (e.g., PCM SSD (Optane™ SSD) and PCM-based main memory (Optane™ DC Persistent DIMM). Thus, we can obtain a similar performance tendency to the current PCM approach. Hence, in this article, we focus on realizing the hybrid main memory system design, based on PCM with DRAM only.

3. Methodology: Q-Selector-Based Prefetching Method for Hybrid Main Memory System

In this section, we introduce the current trends of hybrid main memory systems and its limitations, and the mechanism of our proposed model with its motivation.

3.1. Motivations: DRAM/PCM Hybrid Main Memory System

A general hybrid main memory system is shown in Figure 2a. Due to its industrial maturity and high density with low cost [4], we decided to use PCM as a part of the hybrid memory. The DRAM/PCM hybrid memory systems exploit the heterogeneous features, using the fast DRAM as an off-chip cache. The DRAM cache depends on temporal and spatial locality, but the main memory will still suffer from high DRAM misses because of the low memory locality of modern big-data applications [8]. To overcome this pitfall, we suggest a prefetcher-based hybrid main memory, as shown in Figure 2b. With the prefetcher, the possibility of maintaining hot data in the DRAM increases, which may hide the long latency of the PCM [9]. Additionally, we adopt a small SRAM prefetch buffer to guarantee that the prefetching sequences do not disturb the requests from the processor.

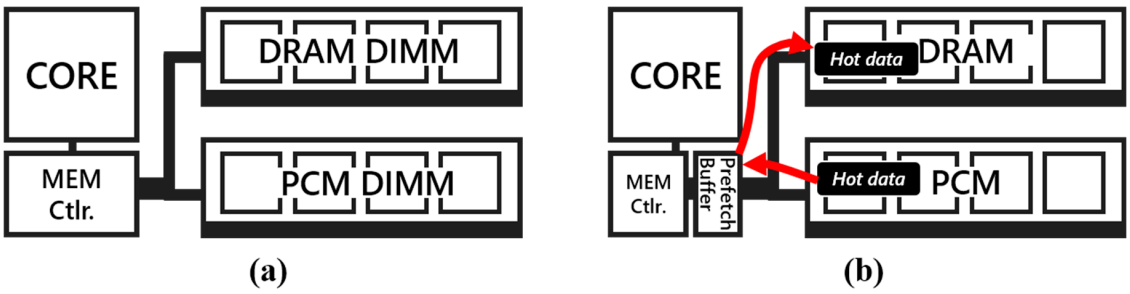

Figure 2. The overall architecture of the hybrid main memory organization: (**a**) the hybrid main memory system with DRAM as a cache and PCM device, and (**b**) the flow of hot data movement from PCM to DRAM cache based on a prefetcher buffer.

Another problem of contemporary applications is that the memory request stream frequently changes its accessing patterns during runtime, as shown in Figure 3b, which shows the memory access

pattern of PARSEC 3.0's streamcluster. At the early phase of its execution, applications construct their data structure by reading the input data, so it shows the stream/sequential memory-access pattern. Consecutively, the program performs the procedure of a core algorithm (e.g., searching the target vertex), showing stride-correlation or pointer-chasing pattern, which is far from following the spatial locality. Thus, the traditional prefetcher, which consists of just one prefetching mechanism, cannot handle this kind of memory access appropriately.

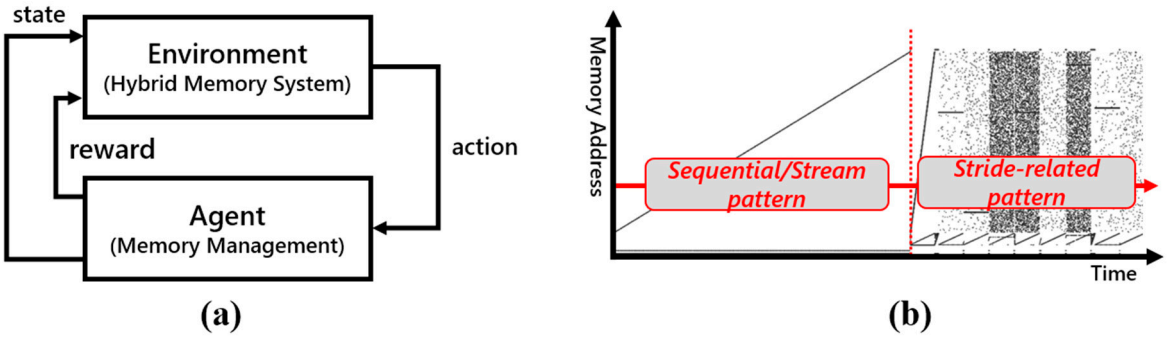

Figure 3. (**a**) The agent-environment interaction on the hybrid memory system based on reinforcement learning method, and (**b**) the visualization result of memory access patterns on a data-intensive workloads, streamcluster from PARSEC 3.0 benchmark suite.

In this paper, we suggest a DRAM-cache-based PCM hybrid main memory system with an intelligent prefetcher based on machine learning, which adopts the Q-learning method for accelerating data mining and big-data applications. To overcome the complexity of prefetching of these big-data applications, our novel method finds the best prefetching method for a certain running phase automatically. For this, we propose a Q-selector-based prefetching (QSBP) method. With QSBP, the applications phase can be classified by the system performance features like current IPC (Instructions per cycle) and LLC (Last-Level Cache) and DRAM-cache hit ratio within a certain interval (e.g., 1k instructions). The reinforcement learning engine learns the tendency and finds the best prefetcher based on given performance features. Our Q-selector method employs the Q-learning technique [10], a well-known reinforcement learning method, which decides its next actions by evaluating the value function with the current state and action pair.

3.2. Reinforcement Learning

Reinforcement learning (RL) [11] is one of the machine-learning methods that can train the best way to achieve a given goal via interaction between an agent and environment as shown in Figure 3a. The RL has been widely adopted to solve various real-world problems, including computer system management. For example, Engin et al. [12] employed Q-learning to automatically choose the memory-scheduling policy, and Kang et al. [13] present an RL-based garbage collection (GC) for an SSD.

Furthermore, reinforcement learning is an online learning method. Hence, it can operate both exploration (gathering more information to approach better decisions) and exploitation (make decisions based on current given information) during online without any pre-training information. So, during its health-checking phase, the system based on the reinforcement learning method may show wrong decisions or inappropriate decisions, however, it can deliver lightweight and more flexible decisions, not fixed by previously trained configurations.

The basic elements of reinforcement learning method are:

- Environment: certain physical environment where the agent operates and interacts,
- State: a set of states of environment and agent,
- Actions: a set of actions of the agent,

- Reward: feedback (rewards or gains after actions by the agent) from the environment,
- Policy: a way to map agent's state to actions,
- Value: future reward (delayed reward) after agent's action at a certain state.

3.3. Design of Q-Selector-Based Prefetching Method

In the case of our hybrid main memory management, the next state and the reward depend on the performance features and the prefetcher pair chosen by the Q-selector. To apply reinforcement learning to our QSBP method for hybrid memory systems, we define four components: states, actions, reward, and policy, with respect to the hybrid memory system. In terms of the actions, we choose one prefetcher from the set of prefetchers, where a stream prefetcher, distance prefetcher, and access-map pattern-matching (AMPM) [14] prefetcher are used. We employ the Q-learning method, managing the Q-values for choosing the best action based on the previous pair. In addition, we set the policy as the ε-greedy method that chooses the action having the maximum Q-value or selects random actions by the given probability function. Figure 4 shows that the next pair of state and action will be determined by the old value of the pair and the learned value.

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\substack{\text{learning} \\ \text{rate}}} \cdot \overbrace{\left[\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\substack{\text{discount} \\ \text{factor}}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right]}^{\text{learned value}}$$

Figure 4. The Q-Learning equation with given parameters.

The Q-value itself tells what extent the next pair <state, action> is appropriate for a near-future environment, so we set the reward function as the hit rate of the DRAM cache during the past 10 k requests. To learn the Q-values, the QSBP uses an equation to continuously update the Q-value table as shown in Figure 5. When QSBP finds the maximum Q-value of the current state's row, it follows the action that has a maximum Q-value. After several iterative refreshes of the Q-values, a choice criterion for the best prefetcher given the environment state (LLC MPKI (miss per kilo instructions), DRAM hit rate) will converge. Therefore, if the current phase follows a high spatial locality, the stream prefetcher and distance prefetcher will be designated for the current state. Otherwise, the current memory-access pattern shows something far different from the regular memory accesses, and the AMPM prefetcher will be selected.

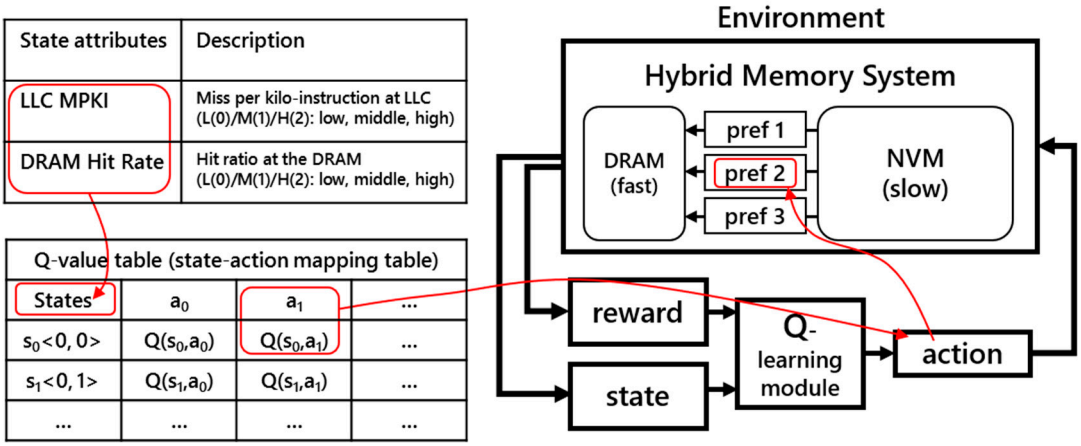

Figure 5. The high-level overview of Q-selector-based prefetching method for hybrid main memory system.

Based on the algorithm shown in Table 2, QSBP determines which prefetcher is the best choice in the current system status (environment). We assume the current memory access pattern's characteristic by evaluating LLC MPKI and DRAM hit rate, refreshing the Q-values of the Q-value table for each LLC miss, and find the maximum value (ε-greedy policy) of the Q-value and select the corresponding action that decides which prefetcher to use.

Table 2. Algorithm for Q-learning-based Q-selector prefetching method.

Inputs: request$_t$, state$_{t-1}$ (S_{t-1}), state$_t$ (S_t), action$_{t-1}$ (A_{t-1})
Output: action$_t$ (A_t)
Learning parameters: α(0.9), γ(0.8)
if LLC_{miss} **then**
$A_t = \max Q(S_{t-1}, \{A_0, A_1, \ldots, A_i\})$
r_t = reward(DRAM-Cache Miss Ratio)
$Q(S_{t-1}, A_{t-1}) = (1 - \alpha) \cdot Q(S_{t-1}, A_{t-1}) + \alpha \cdot \{r_t + \gamma \cdot \max Q(S_t, A_t)\}$
end if

Figure 5 shows that a state is defined by the state attributes consisting of the current system's LLC MPKI and DRAM hit ratio, which is directly related to the current representative status of the DRAM-NVM hybrid main memory. Our method, called QSBP, map current performance state attributes as states and match actions to three different prefetchers. Therefore, determining a prefetcher between three prefetchers will affect system performance, and the feedback refreshes the value of Q-values in the Q-value table to remap the state-action correlation, consequently.

4. Experiments

We evaluate the QSBP model using an in-house x86 architecture simulator that has a similar structure to other memory-level simulators [15,16], which are widely used in the computer architecture research field. We gather user-mode application instructions and memory request traces using Intel Pin [17]. Intel Pin is a dynamic binary instrumentation tool that inserts user-defined instrumentation functions and hander functions without modifying target execution binaries. Hence, we develop an Intel Pin tool library that detects memory and non-memory instructions and collects detailed runtime contexts (e.g., virtual memory address, memory instruction types, etc.) for feeding trace-driven in-house system simulator.

As shown in Table 3, to measure and compare system performance and energy consumption among QSBP and various prefetcher based models, we configure a processor component similar to the Intel® Skylake i7, and other system components including L1/L2/L3 caches, DRAM, NVM, and (hybrid) memory controller.

Table 3. Simulation environment. Memory timings and energies parameters are from [18,19].

Processor	1 core, 4GHz, 4-wide issue
Caches	L1-I and L1-D caches: 32 KiB, 8-ways, 64B/line, LRU replacement L2 Unified cache: 256 KiB, 4-ways, 64B/line, LRU replacement Shared L3 (Last-level cache): 8 MiB, 16-way, 64B/line, LRU replacement
Memory Controller	Memory Scheduler: FCFS scheduling Unified management for DRAM/NVM hybrid configuration
DRAM	Access Latency: 100 ns Energy Consumption: array {read, write} = {1.17, 0.39} pJ/bit
NVM	Cell Device: Phase Change Memory Access Latency: 300 ns (3 times slower than DRAM) Energy Consumption: array {read, write} = {2.47, 16.83} pJ/bit (NOTE: DRAM/PCM size is scaled down to balance in terms of cost)

In addition, as shown in Table 4, we formulate nine single-thread applications from modern memory-intensive benchmarks to evaluate hybrid main memory systems. Target benchmarks are from PARSEC 3.0 [20], and graphBIG [21]; they contain memory-intensive applications and modern big-data analysis and data mining applications that require large working memory space. Additionally, we identify and extract the most representative simulation point (called the region of interest, ROI) to avoid spending much of simulation time on initialization phases and non-critical parts of applications. Further, we scale down DRAM and NVM sizes for each workload to maintain the ratio while considering cost-efficiency.

Table 4. Benchmarks used for the hybrid main memory system workloads.

Benchmark Suite Name	Benchmark Name	Description
PARSEC 3.0 [20]	blackscholes	financial analysis application for calculating prices of options
	canneal	chip optimization tool based on simulated annealing method
	dedup	data-center application which compresses storage images
	ferret	similarity search application based on feature-rich data
	streamcluster	data mining application to solve online clustering problem
graphBIG [21]	bfs	(graph traversal) breadth-first search kernel
	connected component	finding connected subgraphs using BFS traversal
	degree centrality	counting the number of connections linked to vertices
	k-core	finding the minimum value of connection for all vertices
	Graph Data sets	Real world graph data including: Watson Gene Graph, CA RoadNet (California Road Network)

5. Evaluation

We compare four system configurations to measure the benefit of QSBP model against various memory system environments:

- *DRAM only*: a baseline configuration based on conventional DDR4-DRAM main memory,
- *PCM only*: a baseline configuration based on DDR4-PCM main memory without DRAM devices,
- *DRAM/PCM hybrid memory system*: a hybrid main memory-based system based on DRAM/NVM devices without any prefetchers or management policies,
- *DRAM/PCM hybrid memory system with AMPM [14] prefetcher*: a hybrid memory system with AMPM prefetcher,
- *QSBP (proposed)*: our proposed hybrid memory system based on Q-Selector based prefetcher.

Total costs to configure DRAM and PCM memories are the same across all models. We assume that DRAM devices are four times more expensive than PCM devices (4:1).

5.1. Performance

Figure 6 shows the entire execution time of the following models: PCM-only, DRAM/PCM hybrid memory system without prefetcher, DRAM/PCM hybrid memory system with AMPM prefetcher, and our proposed model, QSBP, respectively.

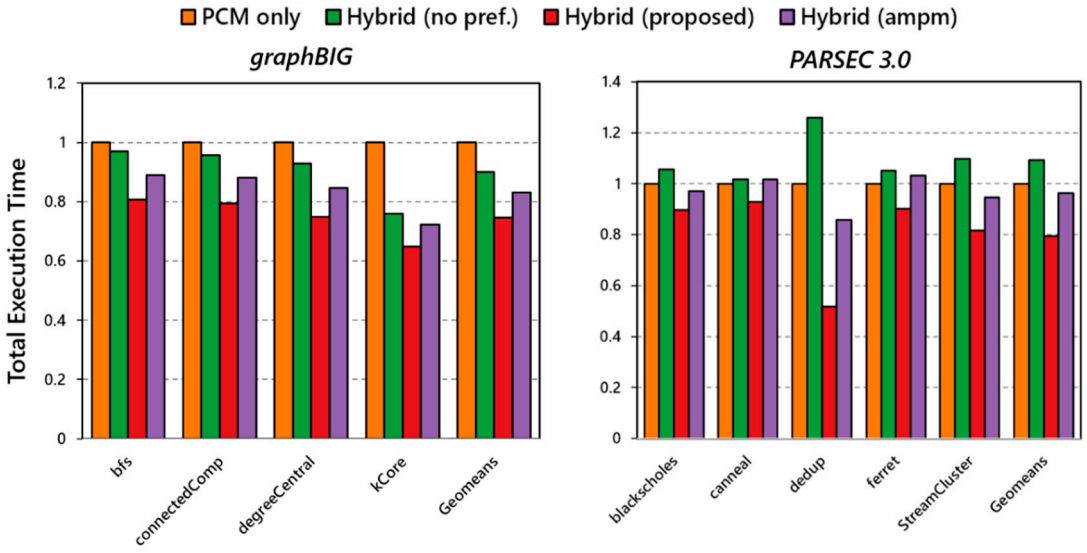

Figure 6. Total execution time of various memory system configurations for graphBIG, and PARSEC benchmarks, normalized to the PCM only main memory system. The figure contains two measurement results from two benchmark suites: graphBIG and PARSEC 3.0.

In this figure, all execution times are normalized to the PCM-only main memory system. Hence, the lower value of execution time than the PCM only model means that the model shows better performance. First, in graphBIG benchmark, all hybrid main memory systems always show reduced execution times over the PCM only model. In the case of geometric means, the proposed model (QSBP) shows approximately 25.4% reduced execution time than the PCM model in graphBIG benchmark, and the other two models also show approximately 10%~17% execution time reduction compared to the PCM only baseline. In addition, the QSBP model achieves 20.4% and 11.3% average execution time reduction compared to the DRAM/PCM hybrid memory system (without any prefetcher), and the hybrid memory system (with AMPM prefetcher) models, respectively. In general, graphBIG benchmarks generate memory requests frequently than conventional scientific applications (e.g., SPEC 2006 or SPEC 2017 benchmark suite [22,23]) and access to a wide region of memory space that is hard to predict by conventional simple prefetchers such as next line prefetcher or stream prefetcher; these two prefetchers are commonly installed on commercial processors [24]. Therefore, introducing prefetchers for hybrid memory systems is one of the promising candidates to avoid memory bottleneck problem when running data-intensive applications.

Consequently, we also measure total execution times for PARSEC 3.0. For this, we choose some workloads from PARSEC, which have memory intensive characteristics and are widely used on modern computing systems such as data center: blackscholes, canneal, dedup, ferret, and streamcluster. As shown in the right side of Figure 6, even DRAM/PCM hybrid memory system (without any prefetcher) configuration degrades its performance compared to the PCM only model, our proposed model (QSBP) achieves 20.6% reduction of execution time compared to the PCM only model. Furthermore, the proposed model shows 27.4% and 17.5% better results than the hybrid (without prefetcher) and hybrid (with AMPM prefetcher) models, respectively. However, in the case of dedup benchmark, the hybrid memory system (without prefetcher) shows a far slower execution time than all other configurations. There are multiple reasons for the increased execution time (the same as reduced performance) for dedup benchmark: (1) dedup is an extremely memory-intensive application, which is used for a data-center environment to compress duplicated disk images (footprints) across multiple virtualized machine instances; and (2) low-ratio of DRAM hit causes high miss penalty (DRAM tag access latency + PCM access + PCM to DRAM migration latency), which can degrade system performance slower than PCM only system without DRAM cache space. The potential pitfall that

just adopts buffer memory spaces or caches for slow memory/storage medium can cause this kind of un-predictable slowdown problem from hidden overheads of cache design.

Hence, other hybrid main memory systems based on prefetchers, such as AMPM-based model and our proposed model, can achieve better performance than the PCM only system on PARSEC benchmark suite, even some irregular workloads have potential threat to cause frequent DRAM misses on hybrid memory architecture.

As shown in Figure 7 and Table 5, the detailed experiment results of performance comparison are provided. The hybrid main memory system without any prefetcher shows 31.38% of energy saving, however, CPI increment (enhancement) degrades 0.25% compared to the PCM-only memory system. The hybrid main memory systems which are based on AMPM prefetcher and the proposed model, QSBP, achieve 79.82% and 64.45% of energy savings, and these two models show 9.81% and 22.70% of CPI enhancement compared to the PCM only main memory system on average.

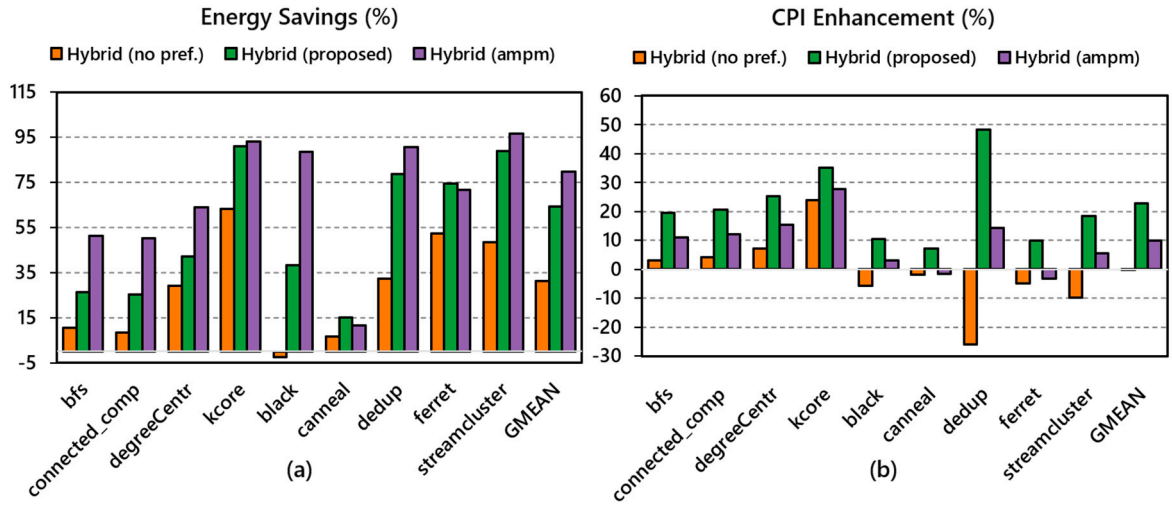

Figure 7. Comparison of various hybrid memory systems based on no-prefetcher, proposed model (QSBP) and AMPM prefetcher, normalized to the PCM-only memory system: (**a**) Energy consumption savings compared with the PCM-only memory system and (**b**) CPI enhancement compared with the PCM-only memory system.

Table 5. Detail simulation results of Figure 7.

	Energy Savings (%)			CPI Increase (%)		
Benchmark	**Hybrid (no pref.)**	**Hybrid (ampm)**	**Hybrid (QSBP)**	**Hybrid (no pref.)**	**Hybrid (ampm)**	**Hybrid (QSBP)**
bfs	10.49%	51.13%	26.38%	3.09%	11.00%	19.38%
connectedComp	8.56%	50.34%	25.46%	4.24%	12.02%	20.50%
degreeCentr	29.27%	64.01%	42.27%	7.23%	15.40%	25.25%
kCore	63.27%	93.01%	90.96%	23.98%	27.73%	35.07%
blackscholes	−2.32%	88.60%	38.22%	−5.59%	3.06%	10.40%
canneal	6.66%	11.49%	14.98%	−1.80%	−1.66%	7.12%
dedup	32.14%	90.70%	78.72%	−26.02%	14.22%	48.32%
ferret	52.50%	71.57%	74.53%	−5.03%	−3.14%	9.84%
streamcluster	48.46%	96.59%	88.88%	−9.86%	5.39%	18.30%
GMEAN	**31.38%**	**79.82%**	**64.45%**	**−0.25%**	**9.81%**	**22.70%**

Moreover, considering I/O waiting time as shown in Figure 8, our QSBP model (proposed) achieved a 33% decrease in running time against the PCM-only model on geometric mean. For this simulation, an additional hardware component is considered to model page fault and disk access.

Hence, target workloads construct large sets of graph structures that exceed working memory size (total capacity of the main memory system); thus, page thrashing and page fault can be huge bottlenecks when running workloads. Therefore, introducing prefetchers to predict demand memory requests that exceed the boundary of working memory size is necessary to satisfy the balance of the entire system performance and the total cost for systems. Even, the hybrid main memory system without prefetcher shows better performance than the DRAM only and the PCM only systems, however, differences are not so adequate to consider an input cost for configuring additional DRAM cache store for PCM memories. Since the hybrid main memory system without a prefetcher cannot exploit the DRAM cache space enough as fast storage for demand request candidates to avoid miss penalty; it uses the DRAM cache space as a storage for recently accessed data. However, hybrid memory systems with prefetchers can store memory data into the DRAM cache space before data are requested by processors, so prefetchers reduce potential DRAM cache miss penalties and can exploit the DRAM space as fast storage for data which could be accessed in near future. On average, the DRAM only model, compared to the base (the PCM only model), provides 4.1% of speedup, when the hybrid main memory system without any prefetcher shows 1% of improvement compared to the base. In addition, the hybrid main memory systems with prefetchers, AMPM, and QSBP, provides an average speedup over the PCM only system of 15% and 33.2%, respectively. These two models are also faster than the DRAM only model, approximately 11.4% and 29.5%, respectively.

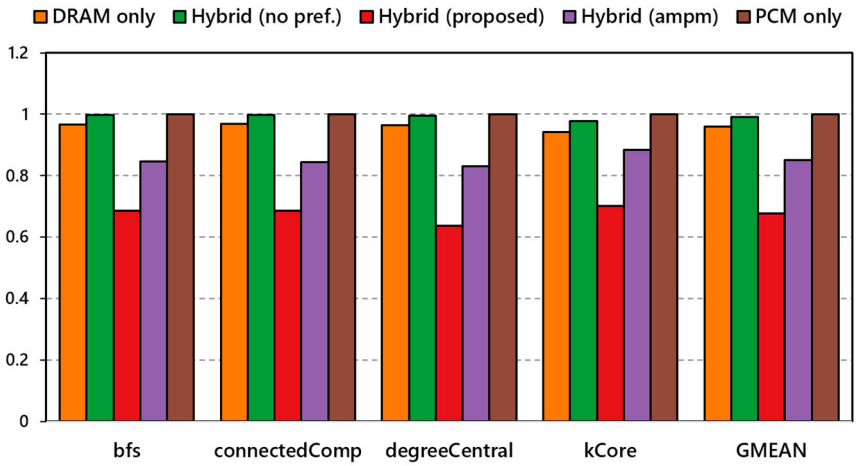

Figure 8. Total execution time with I/O waiting time (considering disk accessing latency) for graphBIG, graph-processing workloads, normalized to the PCM only main memory system.

5.2. Relative IPC

Figure 9 describes the IPC variations among graph workloads and the average of IPC, the PCM only, and the hybrid memory systems with and without prefetchers. Commonly, the IPC represents the total computing performance of a certain system and is not directly related to the memory system power. We observed that our proposed model achieves 25.8%, 37.6%, and 21.1% better IPC compared to the PCM only, the hybrid main memory system without prefetcher, and with AMPM prefetcher, respectively. Hence, we conclude that our proposed method is not sensitively affected by the memory pressure, which is caused by the long latency of PCM devices, compared to other prefetcher-based hybrid memory systems.

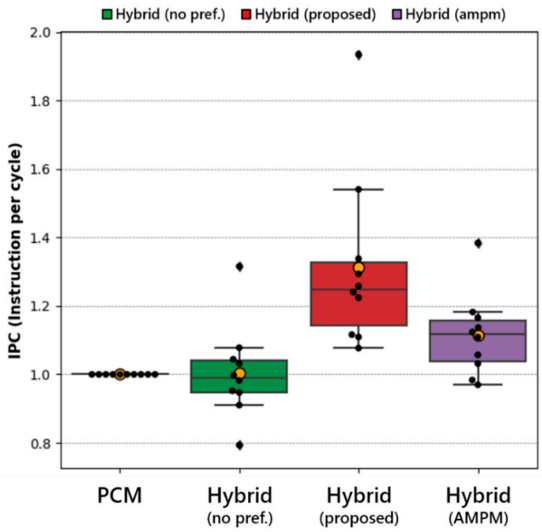

Figure 9. The relative instruction per cycle (IPC) variations of the PCM only system, hybrid memory systems (without prefetcher, with QSBP (proposed)/AMPM prefetcher). The yellow marks show the average relative IPC value of models. All relative IPC values are normalized to the PCM only memory system's IPC values.

5.3. Hit Ratio of DRAM-Cache

Figure 10 shows that our proposed method achieves the better accuracy of hot data placement on DRAM, compared to other prefetch methods. The conventional approach, based on cache-like management DRAM/PCM hierarchy, shows that it cannot determine the appropriate placement of data when running graph-processing and data-intensive benchmarks. In the worst case, the hit rate of the AMPM based hybrid main memory system achieves approximately 19% less hit ratio compared to the proposed model.

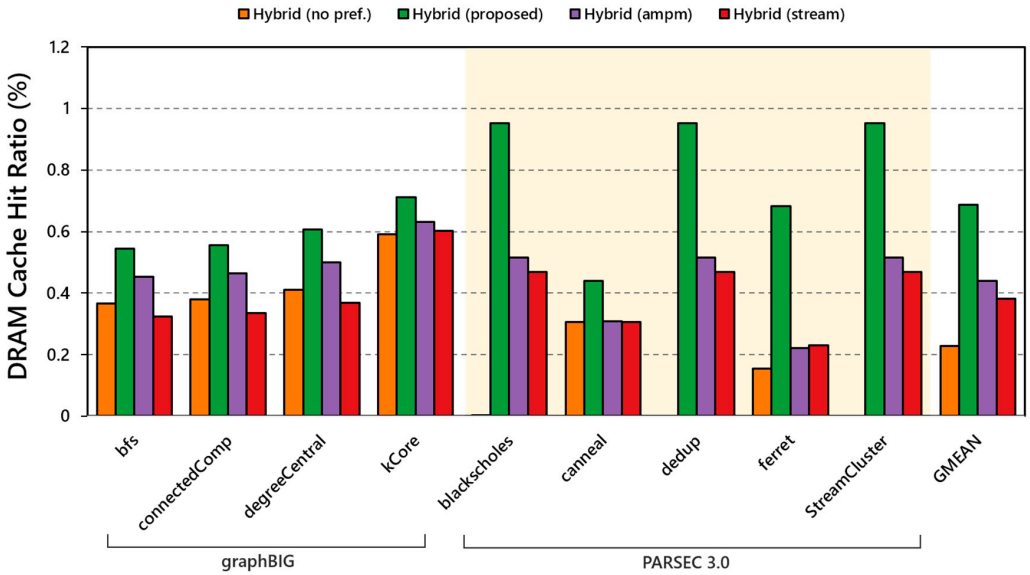

Figure 10. The evaluation results of DRAM cache hit ratio: (1) Hybrid main memory system with no prefetcher, (2) Hybrid memory system with QSBP (proposed model), (3) Hybrid memory system with AMPM prefetcher, and (4) Hybrid memory system with stream prefetcher. The geometric mean (GMEAN) shows overall average value of DRAM cache hit ratio.

Continuously, our proposed model, QSBP, shows approximately 3 times better DRAM-as-a-cache hit ratio compared to the hybrid main memory system without a prefetcher, and shows 55.9%, and 79.5% enhanced hit ratio compared to hybrid memory systems based on AMPM prefetcher and stream prefetcher. Therefore, we conclude that QSBP can generate more accurate prefetcher candidates for hybrid memory systems than other monolithic and fixed functionality prefetchers.

6. Conclusions

In this work, our novel prefetching method, the Q-selector based prefetching method (QBPM), achieves the performance enhancement on the hybrid main memory environment. Traditional memory hierarchy has kept its components as DRAM technologies only past decades, hence, due to the emergence of data-intensive computing, the limitations of DRAM devices act as the performance bottleneck nowadays. Therefore, many researchers have investigated constructing hybrid main memory systems with DRAM and emerging memory technologies such as PCM and STT-MRAM, etc.

However, we observe that existing pitfalls of DRAM-NVM hybrid main memory systems and management methods by system-level simulations: (1) DRAM cache space is inefficiently utilized when there is no additional management for data movement between DRAM and NVM layers, (2) many workloads usually change memory access patterns throughout running time; one monolithic prefetcher cannot handle well these multiple and irregular memory access patterns from modern memory-intensive workloads. Moreover, using DRAM caches without any prefetcher or with an inappropriate prefetcher may cause performance degradation by frequent DRAM cache misses and cache miss penalties. For that reason, we eliminate inefficiencies from hybrid main memory systems by introducing a prefetching mechanism based on a reinforcement learning algorithm. Our proposed model dynamically changes the prefetching strategy by current system performance and memory access pattern characteristics with the small size of the Q-value table and lightweight Q-learning operations.

Using an in-house x86-64 architecture simulator, we measure that the proposed model (QSBP) achieved a 31% performance improvement on average compared to the PCM-only model and delivers an 18% performance gain compared to the existing state-of-the-art prefetcher model. In addition, considering I/O waiting times, we observe that the hybrid main memory system cannot achieve far better performance than the PCM only model as the same amount of construction cost, but the proposed model achieves approximately 47.6% of performance enhancement compared to the PCM only system.

Author Contributions: Conceptualization, methodology, software, J.-G.K.; writing-review and editing, S.-K.Y., and S.-D.K.; supervision, S.-D.K., and S.-K.Y.; funding acquisition, S.-D.K., and S.-K.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea funded by the Korea government (MSIT) (NRF-2019R1A2C1008716) and Research Center for Artificial Intelligence Technology of Jeonbuk National University.

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

Abbreviations

The following abbreviations are used in this manuscript:

NVM	Non-Volatile Memory
PCM	Phase Change Memory
DRAM	Dynamic Random-Access Memory
LLC	Last-Level Cache
QSBP	Q-Selector Based Prefetcher
FCFS	First-Come-First Serve
SSD	Solid-State Drive (Disk)
MPKI	Miss Per Kilo Instruction
IPC	Instructions Per Cycle
RL	Reinforcement Learning

References

1. Qureshi, M.K.; Srinivasan, V.; Rivers, J.A. Scalable high performance main memory system using phase-change memory technology. *ACM Sigarch Comput. Arch. News* **2009**, *37*, 24–33. [CrossRef]
2. Sim, J.; Alameldeen, A.R.; Chishti, Z.; Wilkerson, C.; Kim, H. Transparent Hardware Management of Stacked DRAM as Part of Memory. In Proceedings of the 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture, Cambridge, UK, 13–17 December 2014; pp. 13–24.
3. Qureshi, M.K.; Franceschini, M.M.; Lastras-Montano, L.A. Improving read performance of Phase Change Memories via Write Cancellation and Write Pausing. In Proceedings of the HPCA—16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture, Bangalore, India, 9–14 January 2010; pp. 1–11.
4. Ustiugov, D.; Daglis, A.; Picorel, J.; Sutherland, M.; Bugnion, E.; Falsafi, B.; Pnevmatikatos, D. Design guidelines for high-performance SCM hierarchies. In Proceedings of the International Symposium on Memory Systems, Alexandria, VA, USA, 1–4 October 2018; Association for Computing Machinery (ACM): New York, NY, USA, 2018; pp. 3–16.
5. Zhang, W.; Mazzarello, R.; Wuttig, M.; Ma, E. Designing crystallization in phase-change materials for universal memory and neuro-inspired computing. *Nat. Rev. Mater.* **2019**, *4*, 150–168. [CrossRef]
6. PMDK Library. Available online: https://github.com/pmem/pmdk/ (accessed on 10 December 2020).
7. Boukhobza, J.; Rubini, S.; Chen, R.; Shao, Z. Emerging NVM: A Survey on Architectural Integration and Research Challenges. *ACM Trans. Des. Autom. Electron. Syst.* **2018**, *23*, Article 14. [CrossRef]
8. Wang, J.; Panda, R.; John, L.K. Prefetching for cloud workloads: An analysis based on address patterns. In Proceedings of the 2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Santa Rosa, CA, USA, 24–25 April 2017; pp. 163–172.
9. Islam, M.; Banerjee, S.; Meswani, M.; Kavi, K. Prefetching as a Potentially Effective Technique for Hybrid Memory Optimization. In Proceedings of the Second International Symposium on Memory Systems, Washington, DC, USA, 3–6 October 2016; Association for Computing Machinery (ACM): New York, NY, USA, 2016; pp. 220–231.
10. Sutton, R.S.; Barto, A.G. *Introduction to Reinforcement Learning*, 1st ed.; MIT Press: Cambridge, MA, USA, 1998.
11. Sutton, R.; Barto, A. Reinforcement Learning: An Introduction. *IEEE Trans. Neural Netw.* **1998**, *9*, 1054. [CrossRef]
12. Ipek, E.; Mutlu, O.; Martínez, J.F.; Caruana, R. Self-Optimizing Memory Controllers: A Reinforcement Learning Approach. *Int. Sym. Comput. Arch.* **2008**, *36*, 39–50. [CrossRef]
13. Kang, W.; Shin, D.; Yoo, S. Reinforcement Learning-Assisted Garbage Collection to Mitigate Long-Tail Latency in SSD. *ACM Trans. Embed. Comput. Syst.* **2017**, *16*, 1–20. [CrossRef]
14. Ishii, Y.; Inaba, M.; Hiraki, K. Access map pattern matching for high performance data cache prefetch. *J. Instruct. Level Parallelism* **2011**, *13*, 1–24.
15. Kim, Y.; Yang, W.; Mutlu, O. Ramulator: A Fast and Extensible DRAM Simulator. *IEEE Comput. Arch. Lett.* **2016**, *15*, 45–49. [CrossRef]
16. Champsim. Available online: https://github.com/ChampSim/ChampSim (accessed on 10 December 2020).
17. Luk, C.K.; Cohn, R.; Muth, R.; Patil, H.; Klauser, A.; Lowney, G.; Hazelwood, K. Pin: Building customized program analysis tools with dynamic instrumentation. *Acm Sigplan Not.* **2005**, *40*, 190–200. [CrossRef]
18. Lee, B.C.; Ipek, E.; Mutlu, O.; Burger, D. Architecting phase change memory as a scalable dram alternative. *ACM Sigarch Comput. Arch. News* **2009**, *37*, 2–13. [CrossRef]
19. Yoon, H.; Meza, J.; Ausavarungnirun, R.; Harding, R.A.; Mutlu, O. Row buffer locality aware caching policies for hybrid memories. In Proceedings of the 2012 IEEE 30th International Conference on Computer Design (ICCD), Montreal, QC, Canada, 30 September–3 October 2012; pp. 337–344.
20. Bienia, C.; Kai, L. *Benchmarking Modern Multiprocessors*; Princeton University: Princeton, NJ, USA, 2011.
21. Nai, L.; Xia, Y.; Tanase, I.G.; Kim, H.; Lin, C.Y. GraphBIG: Understanding graph computing in the context of industrial solutions. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Austin, TX, USA, 15–20 November 2015; pp. 1–12.
22. Henning, J.L. SPEC CPU2006 benchmark descriptions. *ACM Sigarch Comput. Arch. News* **2006**, *34*, 1–17. [CrossRef]

23. Panda, R.; Song, S.; Dean, J.; John, L.K. Wait of a Decade: Did SPEC CPU 2017 Broaden the Performance Horizon? In Proceedings of the 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), Vienna, Austria, 24–28 February 2018; pp. 271–282.
24. Farshin, A.; Roozbeh, A.; Maguire, G.Q.; Kostić, D. Make the Most out of Last Level Cache in Intel Processors. In Proceedings of the Fourteenth EuroSys Conference (*EuroSys '19*), Dresden, Germany, 25–28 March 2019; Association for Computing Machinery (ACM): New York, NY, USA, 2019; p. 8.