3

The demand of memory resource for big data applications has increased rapidly in recent years. However, DRAM which has low storage density and high cost cannot meet the ever-increasing memory requirement of many data-centric applications. The advent of Non-Volatile Memory and High-Bandwidth Memory offer new opportunities to address the drawback of DRAM technologies. Compared with DRAM, HBM has higher bandwidth, high storage density, and lower power consumption and NVM has high storage density, low cost per byte, and near-zero static power consumption.

4

Most previous heterogeneous memory systems usually use two kinds of memory media which are organized in a flat or a hierarchical memory architecture. Flat architecture uses both Fast Memory and Slow Memory as main memory. The Fast Memory and Slow Memory are architected in a single address space and managed by the operating system. The hybrid memory controller determines the memory media to access according to address. Hierarchical memory architecture uses Fast Memory as a cache for Slow Memory. Both of them has defects. Flat architecture causes high migration cost and Hierarchical architecture causes high storage cost and complexity of hardware.

5

Our motivation mainly includes three aspects:

First, with the demand of big data and high bandwidth applications for memory, more abundant and multi-layered heterogeneous memory architecture is required. Second, different memory media has different characteristic. It is a great challenge how to make the best memory access performance under the unit cost. Finally, at present, the research of heterogeneous memory architecture is mostly carried out on the simulator, but there is a lack of simulation platform that can support three or more memory media.

7

our multi-tier heterogeneous memory architecture integrates traditional flat and hierarchical memory architectures together, as shown in Figure. In our heterogeneous memory architecture, NVM is used as main memory because it can offer large memory capacity. Since DRAM and HBM are much faster than NVM, they can be used as the filtered cache for NVM. The hotest pages are placed in the HBM, while warm pages are placed in the DRAM. For some memory bandwidth-bound applications, HBM can offer extremely large memory bandwidth to overcome the limitation of DRAM. We implement our multi-tier heterogeneous memory architecture based on GEM5 and DRAMsim3 simulators. It mainly includes three components:

1. CPU cores, TLB, and all levels of cache are modeled with GEM5

2. A hybrid memory controller is designed in the GEM5 simulator to manage HBM, DRAM, and NVM

3. The three memory modules are simulated by DRAMsim3. Both HBM and DRAM are used as the cache of NVM.

8

The hybrid memory controller is the key design of our multi-tier heterogeneous architecture. it manages NVM,DRAM and HBM, including address remapping, memory request dispatching, page hotness monitoring and page migration. It totally contains four major functional modules shown in Figure. In the following, we introduce these modules in detail.

9

Remapping Table is the first module that handles memory access requests in the hybrid memory controller. This module stores the address mappings between NVM, DRAM, and HBM, and thus locates in the critical path of memory accesses. It has three functions. First, this module checks whether a memory request is hit in the remapping table. If an item in the PRT is related to an NVM request, the NVM address should remap to the DRAM or HBM address, and the new request is forwarded to the MRD module. Second, PRT triggers page migration from NVM to DRAM. Third, it updates and maintains the remapping table according to the migration information sent by the MM module.

10

The Dispatcher module is responsible for dispatching memory access requests to different memory media. It also has three major functions. First, it receives the memory access request forwarded by the PRT module and distributes it to NVM, or DRAM/HBM cache. Second, it extracts memory access requests to DRAM and sends them to the PAC module. Finally, it receives page migration requests sent by the MM module and dispatches the corresponding memory requests on demand.

11

The PAC module is responsible for page access counting. It is not on the critical path of memory accesses. The MRD module forwards packets to this module only when a memory request needs to access a DRAM page. The PAC module picks up the hottest pages according to the counters and triggers page migration from DRAM to HBM.

12

The MM module performs page migration requests if a NVM or DRAM page is identified as a hot page. This module is also not on the critical path of memory accesses. When it receives page migration requests from the PRT module or the PAC module, it sends several read and write packets to the MRD module to simulate the page migration between different memory media.

13

The table maintains the address mapping between NVM pages and DRAM/HBM pages. The table shows the data structure of the address remapping table.

The page numbers stored in each entry of the address remapping table can reflect the following two cases. First, if there is only NVM-to-DRAM address mapping, the first two columns of an entry store the NVM page number and the mapped DRAM page number, respectively, and the third column is empty. Second, there is a two-level address mapping, i.e., NVM-to-DRAM and DRAM-to-HBM. Because the hotness of the mapped DRAM page exceeds a given threshold, it should be further migrated to the HBM page. The first two columns of the entry store the NVM page number and the mapped HBM page number, respectively, and the third column stores the original DRAM page number that has been migrated to the HBM page. This design ensures that the second column of the remapping table entry always stores the page number mapped to the NVM page. Each memory access request issued from the bus only needs to look up the remapping table once according to the NVM page number. Compared with the traditional multi-level remapping table, the number of table lookups and the complexity of a single table entry are reduced. Thus, the hardware cost and the performance loss caused by frequent table lookups can be reduced.

14

We exploit the MEA algorithm to identify hot pages in the DRAM cache. As shown in Algorithm 1, MEA is a widely-used sorting algorithm which can sort the hot pages by considering both data access frequency and temporal locality. This algorithm can track dynamical memory access behaviors of applications, and sort hot pages during the execution of program.

16

In this section, we evaluate the multi-tier heterogeneous memory architecture composed of NVM, DRAM, and HBM based on GEM5 and DRAMsim3 simulators.

The performance of hybrid memory system is evaluated mainly from the perspective of bandwidth utilization and Instructions Per Cycle (IPC).

In order to evaluate the accuracy and system performance of the multi-tier heterogeneous memory architecture, five memory systems are set up as follows:

• A pure NVM system: the main memory is only composed of 4 GB NVM

• A pure DRAM system: the main memory is only composed of 4 GB DRAM

• A pure HBM system: the main memory is only composed of 4 GB HBM

• A flat heterogeneous memory system composed of DRAM and HBM: the main memory is composed of 4 GB DRAM and 256 MB HBM

• Multi-tier heterogeneous memory system composed of NVM, DRAM, and HBM: it contains 4 GB NVM main memory, 1 GB DRAM, and 256 MB HBM cache.

We use Standard Performance Evaluation Corporation CPU2017 (SPEC CPU2017) as our benchmarks to evaluate the performance of the above five memory systems.

17

We run the same application on each core for one billions instructions individually, and measure the IPC and bandwidth of applications in five different memory systems. The results are shown in the figure.

In the multi-core execution model, the performance improvement of the flat heterogeneous memory system composed of DRAM and HBM is more obvious than that of the pure DRAM system. For example, when we run the perlbench\_r and x264 r, there are 34.2% and 57.5% performance improvements relative to the pure DRAM system, respectively. The reason is that the memory bandwidth consumption of these applications is rather high, and thus the memory bandwidth of the pure DRAM system becomes the performance bottleneck.

The performance of perlbench\_r and x264 r is even close to the pure HBM system because our page migration scheme can effectively move the hottest data to the HBM, and thus can fully exploit the high-bandwidth feature of HBM.

18

Taking gcc\_r and mcf\_r as examples, we evaluate the impact of the MEA-based hot page monitoring mechanism and the traditional full-count hot page monitoring mechanism on the system IPC in single-core or multi-core scenarios, and the experimental results are shown in this figure.

When we run mcf r for 100 million instructions in the eight core scenario, there is about 7.6% performance gap between the MEA-based hot page monitoring approach and the ideal full-counter approach. For the remaining application scenarios, the performance gap between these two algorithms is less than 1%. This implies our MEA-based approach offers rather high accuracy for hot page monitoring, while significantly reducing the hardware overhead.

19

Our system adopts a random page replacement scheme when there is no free HBM page available. Taking gcc r as an example, we evaluate the application performance under the combinations of different hot page monitoring approaches and different HBM page replacement schemes. The experimental results are shown in the figure. There is little difference (less than 1%) of impact between the random replacement algorithm and the ideal full counter algorithm on the system performance. This implies our random replacement algorithm can achieve comparable system performance while significantly reducing the hardware overhead. In addition, the combination of our random replacement algorithm and MEA-based hot page monitoring algorithm can achieve similar performance compared with the pure HBM system.

21

In this paper, we design a multi-tier heterogeneous memory architecture composed of NVM, DRAM, and HBM, and implement it based on GEM5 and DRAMsim3 simulators. This multi-tier heterogeneous memory architecture uses largecapacity NVM as main memory, and DRAM and HBM which are architected in a flat address space as the cache of NVM. We also design a MEA-based hot page monitor algorithm, a dynamic migration threshold adjustment algorithm, and a random HBM page replacement algorithm to best utilize these heterogeneous memories. Experimental results show that our tiered memory architecture can significantly improve application performance compared with an pure NVM or DRAM architecture, and the performance gap between our HBM/DRAM/NVM architecture and a HBM-only architecture is less than 10%.