THE GEORGE WASHINGTON UNIVERSITY

Computer Architecture and Design
ECE 6005 80

DESIGN AND SIMULATION OF HYBRID MEMORY SYSTEM USING DRAMSIM2

Chethana Ramesh
G44895755

ABSTRACT

- In high performance memory systems , the limitations of DRAM in terms of power consumption is highlighted, where as NVM provides alternative for this problem. This project focuses on designing and simulating a hybrid memory system by using DRAM as Cache and NVM has main memory using DRAMSim2.
- Using DRAMSim2 configured DRAM parameters for providing simulation parameters for real-world memory scenario. Established methods to integrate DRAM as Cache and NVM as main memory for hybrid memory architecture.
- Verified DRAM as cache and demonstrated realistic DRAM latency and power consumption, which provides the base for comparison with hybrid memory.
- In the proposed design it shows the reduction in power consumption, latency and bandwidth by leveraging DRAM as a cache to reduce frequent access and enhanced memory access performance by using NVM's higher write latency.

PROBLEM STATEMENT

- The increase growth in data-intensive applications has enhanced the demand for memory systems that provides balance between the high performance with energy efficiency.
- DRAM is a volatile memory, which means it requires constant power to maintain the stored data, which makes it unsuitable for long-term data storage or energy-efficient systems.
- Traditional DRAM-based systems, provides lower latency and higher bandwidth, consumes more power and face scalability challenges.
- In contrast, emerging Non-Volatile Memories (NVMs) provide increased energy efficiency and scalability but their drawbacks are higher latency and limited write endurance.

MOTIVATION

- This disparity necessitates innovative memory architectures that provides the combination of the strengths of both DRAM and NVM to meet the evolving needs of modern computing systems.
- By utilizing DRAM as a cache for NVM, the hybrid memory system can utilize DRAM's low latency for frequent memory accesses while using NVM as a more energy-efficient, making it suitable scalable solution for less time-sensitive data.
- This approach not only helps balance the trade-off between performance and power but also opens up possibilities for new system designs that can handle the ever-increasing demand for large-scale, energy-efficient computing.

RELATED WORK AND ITS SHORTCOMINGS

1. **"A Case Study of a DRAM-NVM Hybrid Memory Allocator for Key-Value Stores":** S. Volos, T. K. Zhang, J. Gandhi, M. M. Swift, and S. S. Mukherjee, "A Case Study of a DRAM-NVM Hybrid Memory Allocator for Key-Value Stores," *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*, Toronto, ON, Canada, 2017, pp. 527–539, doi: 10.1109/ISCA.2017.59.

- A software-based memory allocator for hybrid DRAM-NVM systems. TARMAC uses DRAM as a cache for NVM and employs intelligent memory management policies to overcome the limitations of hardware-based Memory Mode.

- Evaluations rely on specific hardware configurations (Intel Optane), reducing applicability to environments without such hardware.

- The paper mainly focuses more on performance metrics (throughput and latency) than power consumption, missing the opportunity to explore energy efficiency in hybrid memory systems.

RELATED WORK AND ITS SHORTCOMINGS

2. Z. Zhang, Z. Shen, Z. Jia and Z. Shao, "UniBuffer: Optimizing Journaling Overhead With Unified DRAM and NVM Hybrid Buffer Cache," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 9, pp. 1792-1805, Sept. 2020, doi: 10.1109/TCAD.2019.2925366

- The focus of this work is heavily suitable for database systems and applications that depends up on journaling. Its generalization to broader memory management systems is limited.
- While the approach reduces NVM writes, it doesn't fully address the high write latency inherent in NVM, which can still reduce the performance in write-heavy applications.
- The solution may not scale well for systems with more complex memory access patterns or larger datasets, as the overhead of managing the hybrid buffer can increase with scale.

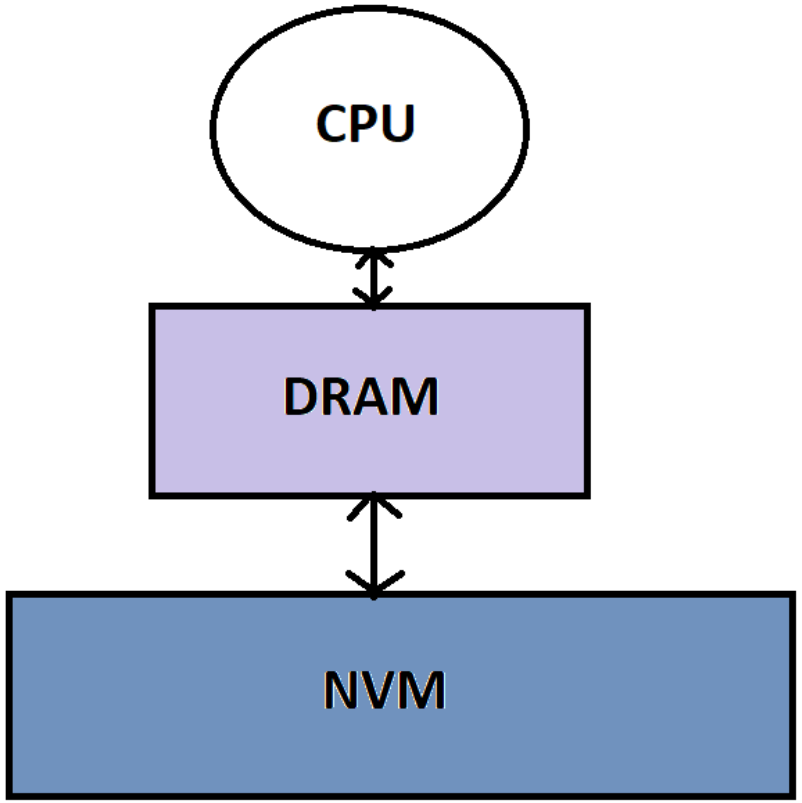
DESCRIPTION OF THE TECHNIQUES PROPOSED

DRAM acts as a Cache for NVM

Concept:
Used DRAM for a high-speed caching layer to store frequently accessed data, while NVM performs as the main memory for long-term data retention.

Benefits:

- Decreases the access latency for hot data by exploiting DRAM's low latency.
- Reduces the power consumption by leveraging NVM's energy efficiency for cold data storage.
- Provides balance between the performance vs. endurance trade-offs of NVM by reducing write amplification through DRAM caching.



DESCRIPTION OF THE TECHNIQUES PROPOSED

Hybrid System Design Framework

Hybrid Architecture:

- The hybrid memory system consists of a CPU connected to DRAM (cache) and NVM (main memory).
- Classes are used for cache (CacheController) and NVM (MainMemoryController) are implemented to manage the data flow and memory access.

Cache Management:

- Implemented caching policies such as Least Recently Used (LRU) and First-In-First-Out (FIFO) to manage the data flow in DRAM.
- Use DRAM as a write-back cache to reduce the frequency of writes to NVM and enhance its endurance.

DESCRIPTION OF THE TECHNIQUES PROPOSED

Data Flow Management

System Operation:

- Memory access requests are directed from the CPU to DRAM for hot data.
- On a cache miss, the request is forwarded to NVM, ensuring seamless data retrieval.

Key Design:

- Bidirectional data flow between DRAM and NVM.
- Synchronization in clock domains for managing timing disparities between DRAM and NVM.

EVALUATION METHODOLOGY

HYBRIDSYSTEM.H

This code integrates DRAM as a Cache and NVM as main memory.

This provides memory access and management.

```
#include "DRAMSim.h"
#ifndef HYBRID_SYSTEM_H
#define HYBRID_SYSTEM_H

#include "CacheController.h"
#include "MainMemoryController.h"

class HybridSystem {
public:
    HybridSystem(MainMemoryController* nvmController, uint64_t cacheSize, uint64_t blockSize);
    bool access(uint64_t address, bool isWrite);  // Handle both cache and NVM access
    ~HybridSystem();

private:
    CacheController* cacheController;  // DRAM cache controller
    MainMemoryController* nvmController;  // NVM controller
    DRAMSim::MultiChannelMemorySystem* memorySystem;  // DRAM memory system

};
```

Image created using data from DRAMSim2. Original analysis

CACHECONTROLLER.H

Handles read or write access.

If data is in cache, it performs operations or moves request to the NVM.

```
#ifndef CACHE_CONTROLLER_H
#define CACHE_CONTROLLER_H

#include <unordered_map>
#include "MainMemoryController.h"

class CacheController {
public:
    CacheController(MainMemoryController* nvmController, uint64_t cacheSize, uint64_t blockSize);
    bool access(uint64_t address, bool isWrite);  // Check cache or forward to NVM
    ~CacheController();

private:
    uint64_t cacheSize;
    uint64_t blockSize;
    std::unordered_map<uint64_t, uint64_t> cache;
    MainMemoryController* nvm;  // Pointer to NVM controller

};
#endif
```

Image created using data from DRAMSim2. Original analysis

EVALUATION METHODOLOGY

MAINMEMORYCONTROLLER.H

Acts as main memory, while integrating DRAM as a cache.

Manages read and write operations, while providing access latency.

```
#ifndef MAIN_MEMORY_CONTROLLER_H
#define MAIN_MEMORY_CONTROLLER_H

#include <cstdint>
#include <unistd.h>
#include <iostream>

class MainMemoryController {
public:
    MainMemoryController(int nvmLatency);  // Constructor to set latency
    bool handleRequest(uint64_t address, bool isWrite); // Handle NVM read/write access
    ~MainMemoryController();  // Destructor

private:
    int nvmLatency;  // Simulated NVM latency
};

#endif
```

SYSTEM.INI

Indicates about the file for performing operations.

Defines about replacement policy, block size, associativity, write policy for cache.

```
workload_trace=k6_aoe_02_short.trc
block_size = 128B
replacement_policy=LRU
associativity=128-way
write_policy=write-back
```

Image created using data from DRAMSim2. Original analysis

Image created using data from DRAMSim2. Original analysis

EVALUATION METHODOLOGY

NVM.INI

NVM has lesser banks with rows and larger bit widths.

NVM does not require refresh cycles and has lower latencies and consumes less power.

```
;
NUM_BANKS=4
NUM_ROWS=16384
NUM_COLS=1024
DEVICE_WIDTH=8


REFRESH_PERIOD=0
tCK=1.5 ;
tRAS=50;  row access strobe
tRCD=30 ; row to column delay
tRP=40  ; row precharge
IDD4W=135; write power
IDD4R=135; read power
```

DRAM.INI

DRAM requires refresh cycles and has higher latencies and consumes more power.

DRAM has higher banks with rows and lesser bit widths.

```
NUM_BANKS=8
NUM_ROWS=32768
NUM_COLS=2048
DEVICE_WIDTH=4

;in nanoseconds


REFRESH_PERIOD=7800
tCK=1.5 ;
tRAS=74;  row access strobe
tRCD=80 ; row to column delay
tRP=60 ; row precharge
IDD4W=255; write power
IDD4R=230; read power
```

Image created using data from DRAMSim2. Original analysis

Captured from the DRAMSim2 simulation tool

RESULTS AND ANALYSIS

Hybrid Memory System

58 transactions at an average aggregate bandwidth of 1.152 GB/s.

Latency varies across various ranks and banks, with the minimum average latency being 39.0 ns and maximum latency reaching 99.375 ns.

Average power consumption range is 1.552 W.

```
   Total Return Transactions : 58 (1856 bytes) aggregate average bandwidth 1.152GB/s
      -Rank   0 :
        -Reads  : 16 (512 bytes)
        -Writes : 0 (0 bytes)
        -Bandwidth / Latency  (Bank 0): 0.079 GB/s              87.000 ns
        -Bandwidth / Latency  (Bank 1): 0.079 GB/s              99.375 ns
        -Bandwidth / Latency  (Bank 2): 0.000 GB/s              -nan ns
        -Bandwidth / Latency  (Bank 3): 0.159 GB/s              61.125 ns
== Power Data for Rank         0
  Average Power (watts)      : 2.026
    -Background (watts)      : 1.043
    -Act/Pre    (watts)      : 0.449
    -Burst      (watts)      : 0.074
    -Refresh    (watts)      : 0.461
     -Rank   1 :
        -Reads  : 4 (128 bytes)
        -Writes : 2 (64 bytes)
        -Bandwidth / Latency  (Bank 0): 0.000 GB/s              -nan ns
        -Bandwidth / Latency  (Bank 1): 0.000 GB/s              -nan ns
        -Bandwidth / Latency  (Bank 2): 0.040 GB/s              -nan ns
        -Bandwidth / Latency  (Bank 3): 0.079 GB/s              39.000 ns
== Power Data for Rank         1
  Average Power (watts)      : 1.494
    -Background (watts)      : 0.999
    -Act/Pre    (watts)      : 0.449
    -Burst      (watts)      : 0.046
    -Refresh    (watts)      : 0.000
     -Rank   2 :
        -Reads  : 16 (512 bytes)
        -Writes : 0 (0 bytes)
        -Bandwidth / Latency  (Bank 0): 0.079 GB/s              60.375 ns
        -Bandwidth / Latency  (Bank 1): 0.000 GB/s              -nan ns
        -Bandwidth / Latency  (Bank 2): 0.079 GB/s              39.000 ns
        -Bandwidth / Latency  (Bank 3): 0.159 GB/s              61.125 ns
== Power Data for Rank         2
  Average Power (watts)      : 1.539
    -Background (watts)      : 1.016
    -Act/Pre    (watts)      : 0.449
    -Burst      (watts)      : 0.074
    -Refresh    (watts)      : 0.000
     -Rank   3 :
        -Reads  : 20 (640 bytes)
        -Writes : 0 (0 bytes)
        -Bandwidth / Latency  (Bank 0): 0.079 GB/s              68.250 ns
        -Bandwidth / Latency  (Bank 1): 0.000 GB/s              -nan ns
        -Bandwidth / Latency  (Bank 2): 0.159 GB/s              57.562 ns
        -Bandwidth / Latency  (Bank 3): 0.159 GB/s              54.750 ns
```

```
== Power Data for Rank         3
  Average Power (watts)      : 1.552
    -Background (watts)      : 1.011
    -Act/Pre    (watts)      : 0.449
    -Burst      (watts)      : 0.092
    -Refresh    (watts)      : 0.000
---  Latency list (6)
      [lat] : #
      [10-19] : 6
      [20-29] : 6
      [30-39] : 13
      [40-49] : 12
      [50-59] : 14
      [60-69] : 5

 == Pending Transactions : 69 (1000)==
//// Channel [0] ////
```

Image created using data from DRAMSim2. Original analysis

RESULTS AND ANALYSIS

DRAM Memory System

27 transactions at an average aggregate bandwidth of 1.073 GB/s.

Higher average latencies in comparison with hybrid memory system, minimum latency is 130.50 ns and maximum is 160.50 ns.

Average power consumption is 3.349W.

```
============== Printing Statistics [id:0]==============
  Total Return Transactions : 27 (1728 bytes) aggregate average bandwidth 1.073GB/s
     -Rank   0 :
       -Reads  : 25 (1600 bytes)
       -Writes : 2 (128 bytes)
       -Bandwidth / Latency  (Bank 0): 0.000 GB/s             -nan ns
       -Bandwidth / Latency  (Bank 1): 0.040 GB/s             160.500 ns
       -Bandwidth / Latency  (Bank 2): 0.000 GB/s             -nan ns
       -Bandwidth / Latency  (Bank 3): 0.000 GB/s             -nan ns
       -Bandwidth / Latency  (Bank 4): 0.079 GB/s             -nan ns
       -Bandwidth / Latency  (Bank 5): 0.318 GB/s             156.375 ns
       -Bandwidth / Latency  (Bank 6): 0.318 GB/s             130.500 ns
       -Bandwidth / Latency  (Bank 7): 0.318 GB/s             136.125 ns
== Power Data for Rank        0
  Average Power (watts)     : 3.349
    -Background (watts)     : 2.405
    -Act/Pre    (watts)     : 0.438
    -Burst      (watts)     : 0.506
    -Refresh    (watts)     : 0.000
---  Latency list (7)
      [lat] : #
      [60-69] : 1
      [70-79] : 4
      [80-89] : 5
      [90-99] : 10
      [100-109] : 2
      [130-139] : 1
      [140-149] : 2

== Pending Transactions : 37 (1000)==
```

Image created using data from DRAMSim2. Original analysis

RESULTS AND ANALYSIS

Bandwidth:

The hybrid system achieves **1.152 GB/s**, which is higher than the DRAM system's **1.073 GB/s**, showcasing its ability to handle memory requests more efficiently by utilizing both DRAM and NVM.

Latency:

Hybrid memory outperforms DRAM-only in terms of average latency:

- Hybrid: Minimum latency is **39.0 ns**.
- DRAM-only: Minimum latency is **130.50 ns**, showing a less consistent performance under load.

Power Consumption:

The hybrid system consumes **very less power** than the DRAM-only system:

- Hybrid: **1.552W** average.
- DRAM-only: **3.349W**, driven primarily by high background power.

FUTURE WORK

Workload Diversity and Trace Simulation

- Testing of hybrid memory system with a various workloads, including read-heavy, write-intensive, and mixed scenarios.
- Using real-world traces such as SPEC benchmarks, database applications for comprehensive testing.
- Evaluating how the system handles dynamic memory access patterns, such as streaming data or irregular memory operations.

Energy Efficiency Enhancements

- Investigating different techniques to reduce idle power consumption in DRAM and NVM.
- Implementing memory schedulers that optimize energy efficiency while maintaining performance.

CONCLUSION

- The project demonstrates the integration of DRAM as a cache and NVM as main memory using DRAMSim2, highlighting the feasibility of hybrid memory systems.
- The hybrid memory system has achieved lower latencies for frequent accesses in comparison to DRAM-only configurations due to DRAM's high-speed caching. Power consumption in the hybrid setup was significantly reduced.
- The hybrid system can handle more transactions, showcasing better efficiency in managing memory requests than the DRAM-only configuration. Bandwidth utilization was slightly improved in the hybrid setup.
- Hybrid DRAM-NVM systems shows strong potential for energy-efficient, high-performance computing, suitable for applications like data centers, cloud computing, and embedded systems.

THANK YOU