Implementation of a Heterogeneous-Reliability Memory Framework

Konstantinos Tovletoglou
Queen's University Belfast, United Kingdom
Supervisors: Georgios Karakonstantis and Dimitrios S. Nikolopoulos

1 Introduction

DRAM consumes up to 40% out of the total power dissipation in servers [2]. Techniques try to reduce it by relaxing the DRAM timing parameters, such as refresh rate. However operation under relaxed parameters poses a threat to the reliability. To allow the operation of the system even when errors occur, recent schemes [3] suggest the usage of heterogeneous-reliability memory. Critical data are protected in reliable memory, while the non-critical data are stored in unreliable memory with relaxed parameters that is more energy efficient.

Our aim is to develop a heterogeneous-reliability DRAM framework and implement it on a commodity server. The implementation of such a framework faces three major problems: *i)* the existence of hardware-based memory interleaving, implemented in all server memory controllers (MCUs), which does not allow to differentiate the address space of the two memory domains, reliable and unreliable, *ii)* if the interleaving is disabled, a solid performance overhead is introduced and finally *iii)* an easy to use interface does not exist for users to adopt this technology.

2 Proposed Implementation

Our experimental setup is based on the *AppliedMicro X-Gene 2* processor [4], which has 4 DDR3 MCUs operating at 1866 MT/s populated with 4×8GB DIMMs. The MCUs interleaving can be controlled from the firmware.

With interleaving enabled, allocations are spread uniformly across all the MCUs. If we relax the parameters of one MCU, we cannot ensure the reliability of critical data as they are spread in all MCUs. We disable the interleaving so that each MCU has a distinct address space and allocations are targeting a single MCU.

The Linux OS and the interface of Non-Uniform Memory Access (NUMA) are modified to support the heterogeneous-reliability memory without requiring any modifications in the hardware. Our framework provides the ability to *i)* reliably allocate critical data in MCUs with nominal parameters and *ii)* enable the use of unreliable storage for the rest of the data in MCUs that have relaxed parameters. Our development modifications are generic and can be applied to any server with multiple MCUs with separate address spaces.

The default policy is to execute application under reliable memory. We expose the two reliability domains as NUMA domains, so the programmer may force an application to use only unreliable memory, through the `numactl` interface. For finer control, the source code of the application can be modified and typical dynamic allocations, `malloc`, can be converted to `numa_alloc_onnode` in which you can specify the reliability domain on which the allocation happens.

We are executing 6 NAS benchmarks [1] and we measure the performance overhead introduced by disabling the interleaving in the system, which can reach as high as $1.49\times$ and on average 63% compared to the baseline hardware-based interleaved system (Figure 1a).

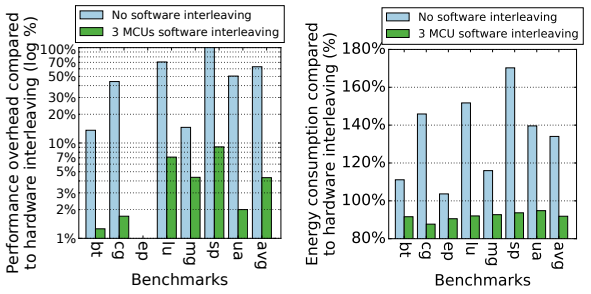

Figure 1: Comparison of *a)* the performance overhead and *b)* the total energy, between the non-interleaved and the software-based interleaved system normalized to the hardware-based interleaved system.

3 Software-based Interleaving

We propose a solution to alleviate the performance overhead of such a framework by utilizing software-based interleaving of the MCUs. We utilize the NUMA interface to map physical pages from different MCUs to continuous virtual addresses for the application. We can interleave dynamically the memory allocations through the `numactl` interface, allowing us to spread allocations on multiple MCUs and choose which bits of the virtual address define the interleaving.

The software-based interleaving limits the performance overhead to 6.3% on average and 9% in the worst-case out of the 6 NAS benchmarks compared to the hardware-based interleaved system (Figure 1a).

By relaxing the refresh rate by $35\times$ and the voltage by 5% of the unreliable domain, we achieve up to 27.6% DRAM power reduction and on average 19.9%. Figure 1b shows the total energy of the system which is reduced by 8.8% on average while taking into account the incurred performance loss.

In future work, we will further investigate the interleaving policies [5] which can be dynamically adjusted to improve performance and we will port popular workloads, such as in-memory databases, to our heterogeneous-reliability framework.

References

[1] D. Bailey et al. The nas parallel benchmarks. *IJSA '91*.
[2] B. Giridhar et al. Exploring dram organizations for energy-efficient and resilient exascale memories. *SC '13*.
[3] S. Liu et al. Flikker: Saving DRAM Refresh-power Through Critical Data Partitioning. *ASPLOS '11*.
[4] G. Singh et al. AppliedMicro X-Gene 2. *Hot Chips '14*.
[5] Z. Zhang et al. A permutation-based page interleaving scheme to reduce row-buffer conflicts and exploit data locality. *MICRO '00*.