Implementation of a Heterogeneous-Reliability Memory Framework

Implementation of a Heterogeneous-Reliability Memory Framework

Konstantinos Tovletoglou
Georgios Karakonstantis and Dimitrios S. Nikolopoulos

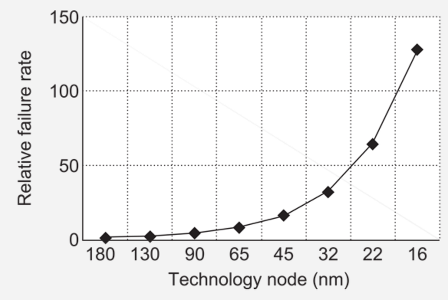
Institute of Electronics, Communications and Information Technology (ECIT)
Queen's University Belfast, United Kingdom

QUEEN'S UNIVERSITY BELFAST

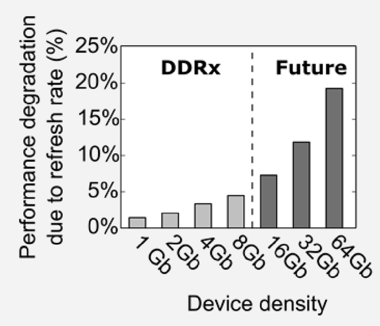
ECIT THE INSTITUTE OF ELECTRONICS COMMUNICATIONS AND INFORMATION TECHNOLOGY

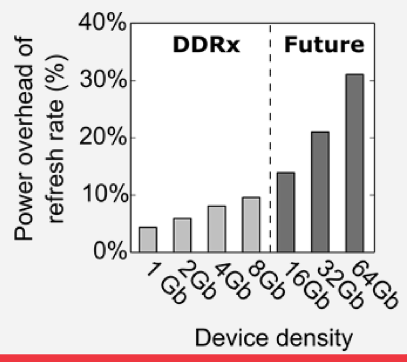
Motivation

- Nanometer memories are becoming unreliable
  - ➤ Increased failure rates threatening the system



- Conventional approach: adoption of guardbands based on the worst-case scenario
  - ➤ Power and performance overhead
- DRAM consumes up to 40% of the total power dissipation in servers

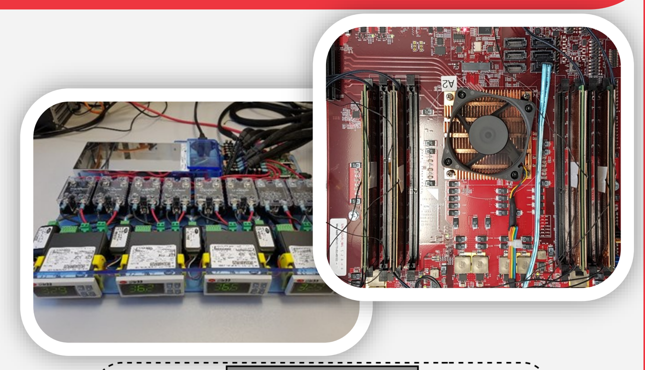




Experimental Setup

Implemented on a real commodity server

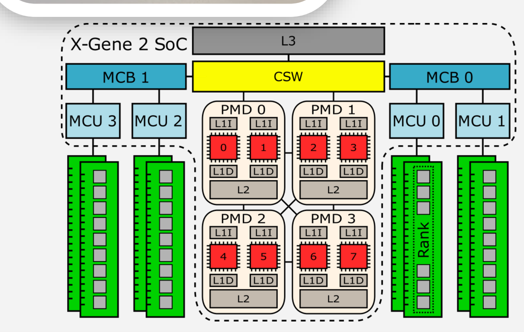
- AppliedMicro X-Gene 2, 8 × AArch64 cores
- 4 Memory controllers (MCUs), 4 × DIMM DDR3 8GB
- CentOS 7, Linux kernel 4.11



Evaluated with 35 workloads (SPEC CPU2006 and NAS)

Parameters of the variably-reliable memory domain:

- Refresh rate: 35x relaxed (64 ms to 2.283 s)
- Voltage: 5% reduction (1.5 V to 1.425 V)



Proposed Approach

Heterogeneous-Reliability Memory Framework (HRM)

Separate the memory into two domains and allocate data on each one based on their criticality and tolerance to errors.

Reliable memory domain	Variably-reliable memory domain
• High cost guardbands	• Relaxed DRAM parameters
Storage of:	Storage of:
• Critical data	• Error-resilient data

- Existing approaches showcased:
  - the potential gains of HRM on simulators
  - identified the existence of variable criticality of application data

Challenges

- × Evaluated only on simulators
- × The existence of hardware-based memory interleaving
- × Disabling interleaving introduces a performance overhead
- × The lack of an intuitive interface for the HRM

Proposed HRM

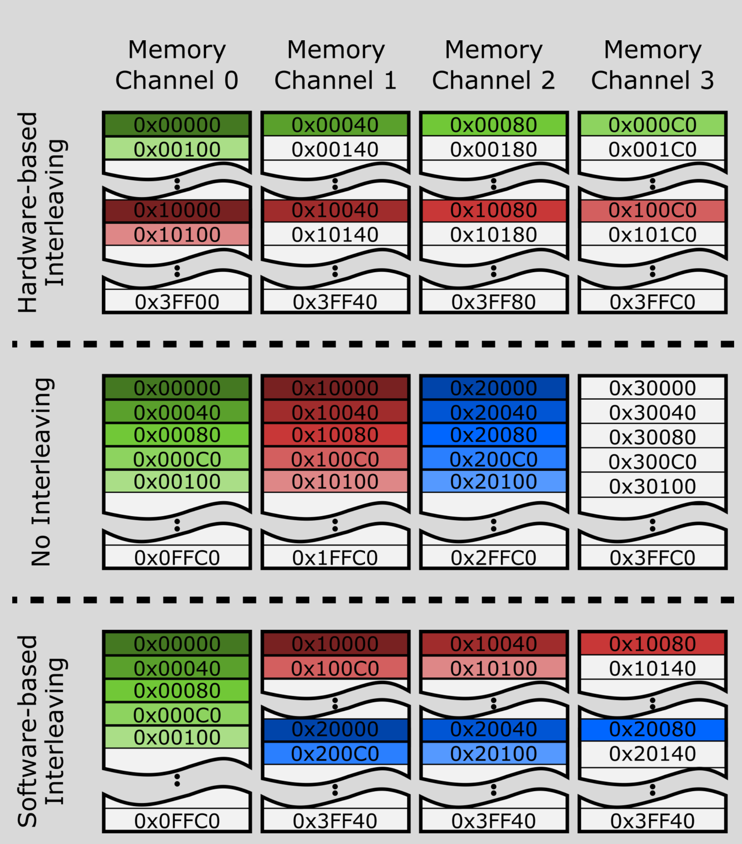
Interleaving

Expose and disable the hardware-based memory interleaving on the server

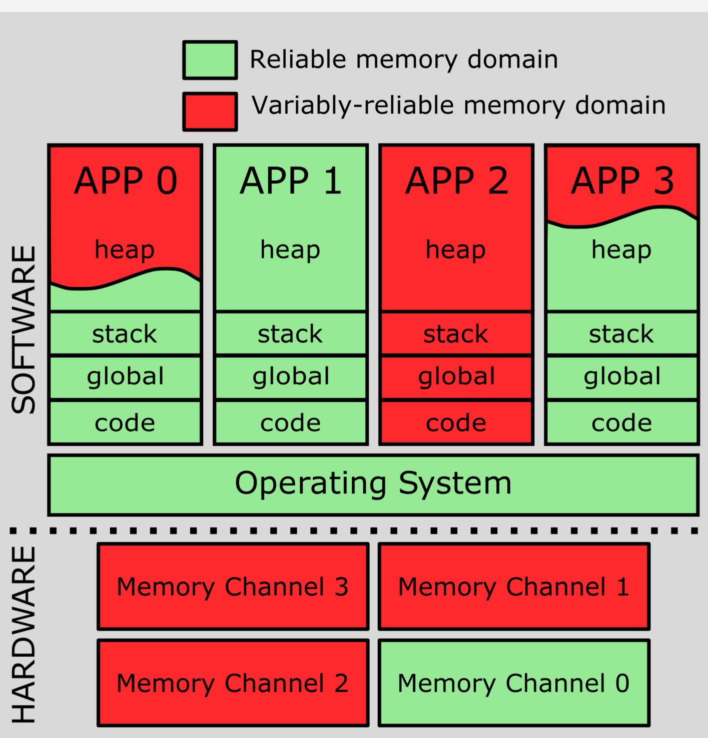
- Enable distinct memory address ranges for each memory channel
- Performance overhead is introduced

Implement a software-based memory interleaving scheme

- Exploit multiple memory controllers for consecutive accesses
- On-the-fly selection of the interleaving function



Interface



Introduce an interface for HRM allocations under the Linux OS

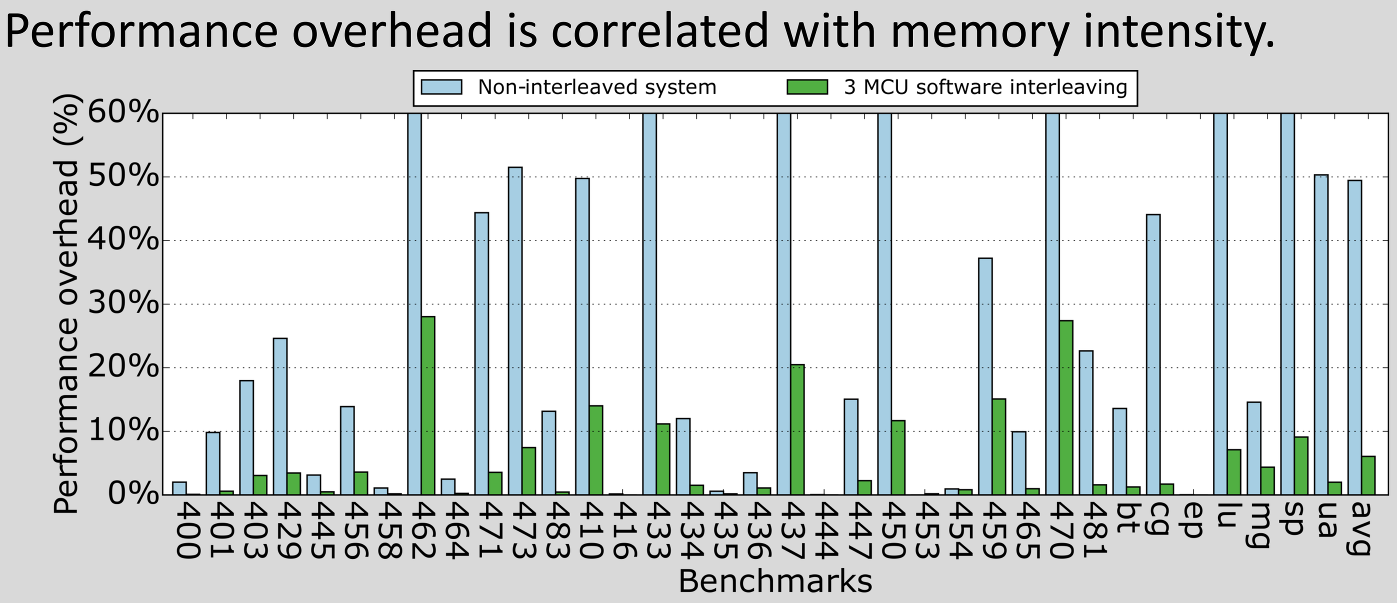
- NUMA interface, `numactl`, to control on application-level (e.g. APP1, APP2)
- Allocation functions, `malloc`, can be replace with `numa_alloc_onnode`, to specify the reliability domain for each allocation (e.g. APP0, APP3)

Enable the selection of software-based interleaving through the same interface

Experimental Results

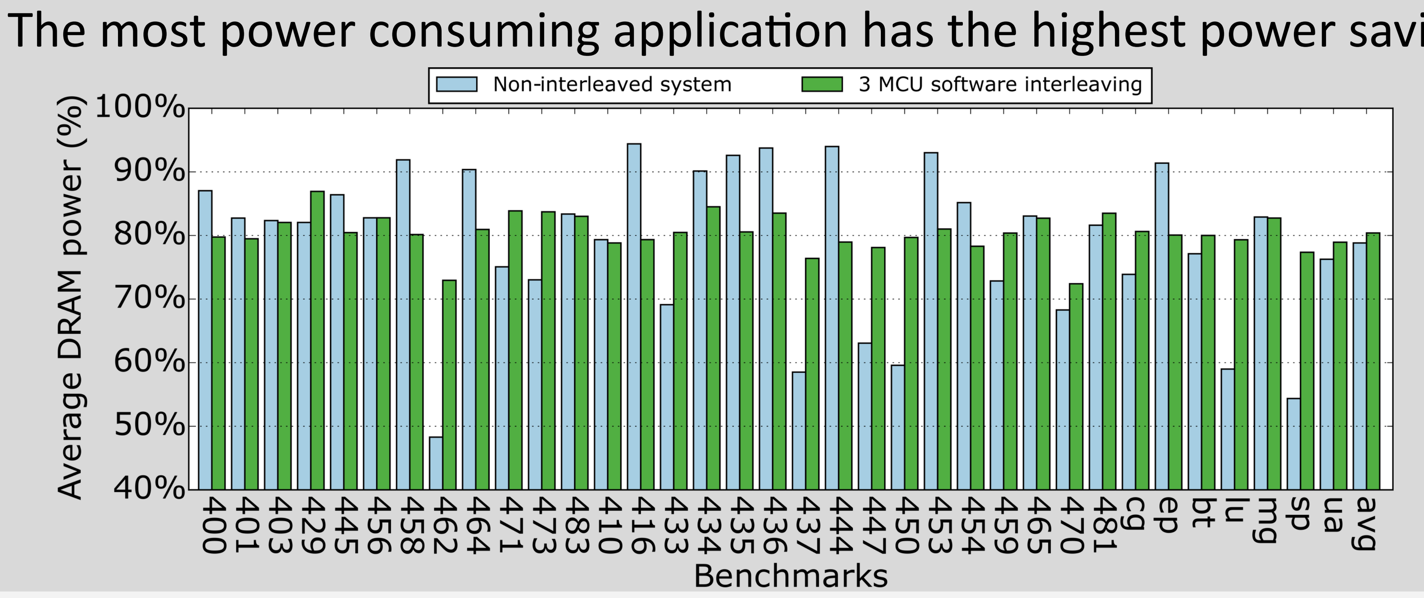
Performance

- The naive HRM introduces an average performance overhead of 49% and it reaches up to 128% for `462.libquantum`.
- Our implementation **decreased the average overhead down to 6%** , while `462.libquantum` has the highest overhead at only 28%.
- Performance overhead is correlated with memory intensity.



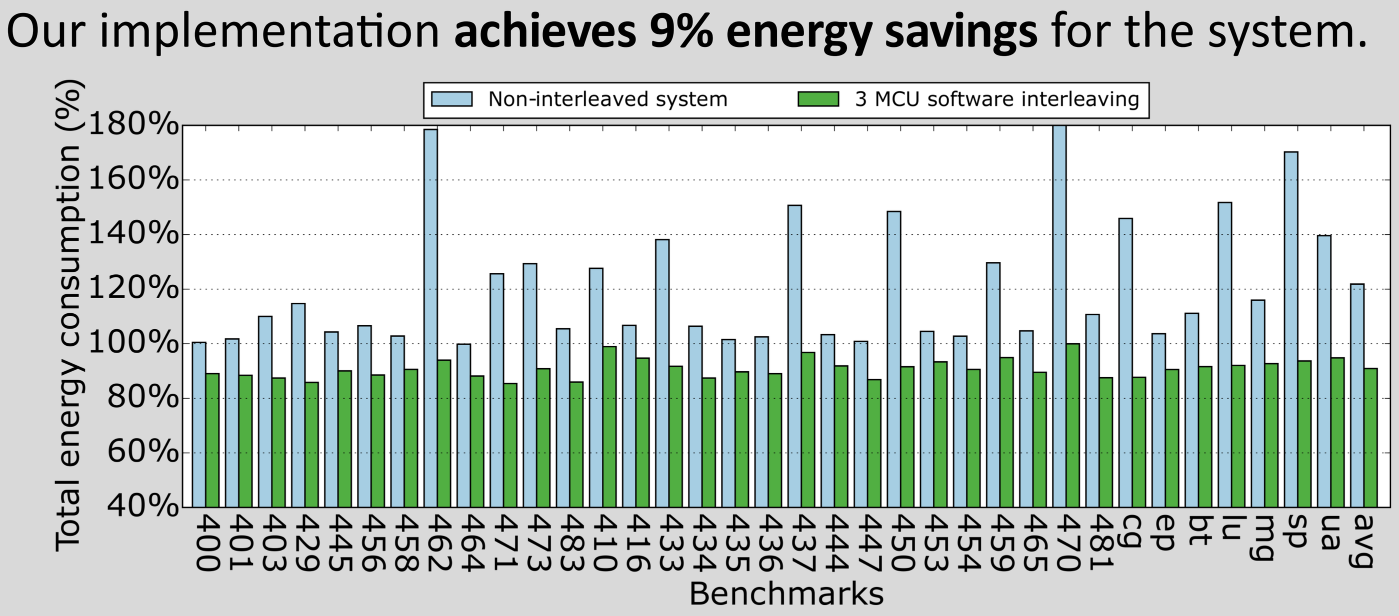
Power

- The naive HRM decreases the power consumption by 23%.
- Our implementation **reduces the DRAM power consumption by 20%**.
- The most power consuming application has the highest power savings.



Energy

- For the naive HRM, no benchmark achieves any energy savings, and the energy of the system (processor and DRAM) is increased by 22%.
- Our implementation **achieves 9% energy savings** for the system.



Reliability

- Under non-controlled temperature, only correctable errors occur in the variably-reliable memory domain, while under high temperature, uncorrectable errors manifest and applications must tolerate them.
- No errors occur in the reliable domain even at high temperature.

Conclusions

- Implement a heterogeneous-reliability memory framework on a real server.
- Introduce a software-based interleaving technique to mitigate the performance overhead when hardware-based memory interleaving is disabled.
- Obtain 9% energy savings and reduce DRAM power consumption by 20%.
- Enable fine-grain control of the allocation on the reliability domains.
- Ensure that errors will not manifest in the critical data, such as OS data.

Contact email: ktovletoglou01@qub.ac.uk

H2020 programme : UniServer (grant no. 688540) : www.uniserver2020.eu