Implementation of a Heterogeneous-Reliability Memory Framework


Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

Publisher rights
Copyright 2018 The author.

General rights
Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.

Download date: 30. Jun. 2020
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

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 x AArch64 cores
  - 4 Memory controllers (MCUs), 4 x DIMM DDR3 8GB
  - CentOS 7, Linux kernel 4.11
- Evaluated with 35 workloads (SPEC CPU2006 and NAS)

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 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
  - High cost guardbands
  - Storage of:
    - Critical data
  - Existing approaches showcased:
    - the potential gains of HRM on simulators
    - identified the existence of variable criticality of application data
- Variably-reliable memory domain
  - Relaxed DRAM parameters
  - Storage of:
    - Error-resilient data

Proposed HRM
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

Introduce an interface for HRM allocations under the Linux OS
- NUMA interface, numacl
  - to control on-application-level (e.g. APP1, APP2)
- Allocation functions, numa_alloc_onnode, can be replace with numa_a11oc_omnode, to specify the reliability domain for each allocation (e.g. APPD, APP3)

Experimental Results
- 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.

Reliability
- 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.

Energy
- 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.

Conclusions

Contact email: ktovletoglou01@qub.ac.uk
H2020 programme : UniServer (grant no. 688540) : www.uniserver2020.eu