Thesis Overview:

Write management mechanisms for systems with non-volatile memory technologies

Roberto Alonso Rodríguez Rodríguez
Computer Science Faculty, Complutense University of Madrid, Spain
PhD Thesis in Computer Science[1]
Advisors: Fernando Castro and Daniel Chaver
{rrodriguezr,fcastror,dani02}@ucm.es

Since the beginning of computer systems, the memory subsystem has always been one of their essential components. However, the different pace of change between microprocessor and memory has become one of the greatest challenges that current designers have to address in order to develop more powerful computer systems. This problem, called memory gap, is further compounded by the limited scalability and the high energy consumption of conventional memory technologies (DRAM and SRAM), which has leaded to consider *new non-volatile memory* (NVM) technologies as potential candidates to replace them. Among NVMs, PCM and STT-RAM [1] are currently postulated as the best alternatives.

Although PCM and STT-RAM have significant advantages over DRAM and SRAM, they also suffer from some drawbacks that need to be mitigated before they can both be employed as memory technologies for the next computers generation. Notably, the slow and energy-hungry write operations on both technologies, and the limited endurance of PCM cells, which become unchangeable after performing a relatively reduced amount of writes on them, are the main constraints of PCM and STT-RAM technologies. This thesis presents two proposals aimed to efficiently manage the write operations on this kind of memories.

The first proposal, conceived for a system with a PCM-based main memory, is intended to reduce the number of writes to the main memory by operating at the cache controller level through the replacement policy used in the immediate-lower memory hierarchy level (the last-level cache, LLC). For this purpose, and as the starting point, the conventional LLC replacement policies (oriented to improve the system performance) have been evaluated in terms of the amount of writes generated to main memory. Notably, in this work we explored the behavior of LRU [2], peLIFO [3], SHIP [4], SRRIP and DRRIP [5] policies. Once the algorithm reporting the lowest amount of writes to main memory has been identified (DRRIP), several changes are proposed aimed to find a replacement policy satisfying the twofold goal of minimizing the number of writes to PCM main memory (and hence reducing the corresponding energy consumption and extending the PCM lifetime) and not penalizing the system performance. More specifically, the proposed algorithms modify the insertion, promotion and also the victimization sub-policies of DRRIP in order to maintain dirty blocks in the LLC. The proposed algorithms have been encoded and integrated in the *gem5* architectural simulator, so that the desired environment, where the main memory is modeled according to PCM memory features and the last-level cache operates with the explored replacement policies, is simulated. The behavior of these proposed algorithms when running different kinds of applications, both sequential and parallel programs as well as multiprogrammed workloads, is evaluated. Experimental results show [6,7] that, on average, compared with a conventional LRU algorithm, some of our proposals manage to extend the memory lifetime up to 20–45%, also reducing the energy consumption in the memory hierarchy by up to 9% and hardly degrading performance.

In the second proposal, conceived for a system with an STT-RAM last-level cache, a mechanism aimed to predict unnecessary writes to this last-level cache is presented, so that those writes identified as useless are filtered in the LLC and performed directly in the main memory. For this purpose, it was first explored the reuse locality [8] that the stream of references arriving at the LLC exhibits, unlike the temporal locality that exhibits the stream of references arriving to the cache levels closer to the processor. Once verified and evaluated this feature, it was exploited by using an element which is able to detect those blocks exhibiting reuse. This reuse detector [9] is in charge of managing the LLC contents, so that the blocks predicted to be non-dead blocks are inserted in the LLC while those predicted to be dead (i.e. non-reused blocks) bypass the LLC, hence reducing the amount of writes to this level and also the corresponding energy consumption. For the evaluation of this approach, the reuse detector, as well as the required modifications in order to adapt the coherence mechanism, were encoded using the *gem5* architectural simulator, where also the LLC was modeled according to STT-RAM memory features. Then the proposal was evaluated using sequential applications and multiprogrammed

[1] Full text available at: http://repositorio.conicit.go.cr:8080/xmlui/handle/123456789/207

workloads in a multiprocessor environment. Experimental results reveal that this content management technique, applied to an STT-RAM LLC and compared to an STT-RAM LLC baseline where no reuse detector is employed, reports energy reductions in the shared LLC of a multiprocessor system of around 40%, an additional energy reduction of more than 6% in the main memory, and improves performance by 3-7% depending on the specific features of the different multiprocessor systems evaluated. Also and more importantly, our scheme is able to outperform DASCA, the state-of-the-art STT-RAM LLC management scheme [10], reporting LLC energy savings of 5-10% more than DASCA and higher performance improvements (in the range 2-9%) depending on the scenario evaluated.

Finally, in this work we have also developed a methodology for building the Miss-Rate Curve (MRC) of an application [11]. The MRC reports an application's cache occupancy on a given cache level (usually the shared LLC in a multi-core scenario) vs. a certain metric related with performance, such as the number of Misses Per Kilo Instructions (MPKI). Overall, our technique works as follows: By using PMCTrack [11] (an OS-oriented PMC tool for the Linux kernel) we periodically gather the MPKI and, by making use of Intel's CMT [12], we collect the LLC occupancy of the co-running applications, thus obtaining different discrete MRC points. Note, however, that when several applications share a cache, they may reach an equilibrium state in the distribution of the cache. In that case, in order to obtain points in the whole range of cache sizes, we slow down co-runner applications by applying duty-cycle modulation techniques to the cores where they run. This allows other applications to increase their occupancy, which in turn, makes it possible for us to explore different MPKI values for the whole cache size range. Then, when enough points have been collected, we apply regression analysis to obtain the whole MRCs for the applications.

In conclusion, we must highlight that it is possible to design architectural solutions that mitigate the shortcomings of NVMs and facilitate their route to become the natural replacement of the exhausted conventional technologies. By addressing this issue at different levels, it has been shown that PCM and STT-RAM may be efficient alternatives to the usage of DRAM and SRAM as technologies of the main memory and the last-level cache, respectively.

References

[1] M. K. Qureshi, S. Gurumurthi, and B. Rajendran, "Phase change memory: from devices to systems", Synthesis Lectures on Computer Architecture, vol. 6, no. 4, pp. 1–134, 2011.

[2] JL Hennessy and DA Patterson, Computer architecture: a quantitative approach. Morgan K. Pub, 2011.

[3] M. Chaudhuri, "Pseudo-LIFO: the foundation of a new family of replacement policies for last-level caches", in IEEE/ACM International Symposium on Microarchitecture (MICRO), 2009, pp. 401–412.

[4] C.J. Wu, A. Jaleel, W. Hasenplaugh, M. Martonosi, S. C. Steely, and J. S. Emer, "SHiP: signature-based hit predictor for high performance caching", in Int. Symposium on Microarchitecture, MICRO 2011, pp. 430–441.

[5] A. Jaleel, K. B. Theobald, S. C. Steely, and J. S. Emer, "High performance cache replacement using re-reference interval prediction (RRIP)", in Int. Symposium on Computer Architecture (ISCA), 2010, pp. 60–71.

[6] R. Rodríguez-Rodríguez, F. Castro, D. Chaver, L. Piñuel, and F. Tirado, "Reducing writes in phase-change memory environments by using efficient cache replacement policies", in Design, Automation and Test in Europe (DATE), 2013, pp. 93–96.

[7] R. Rodríguez-Rodríguez, F. Castro, D. Chaver, R. Gonzalez-Alberquilla, L. Piñuel, and F. Tirado, "Write-aware replacement policies for PCM-based systems", The Computer Journal, 58 (9), pp. 2000–2025, 2015.

[8] J. Albericio, P. Ibáñez, V. Viñals, and J. M. Llabería, "Exploiting Reuse Locality on Inclusive Shared Last-level Caches", ACM Trans. Archit. Code Optim., vol. 9, no. 4, 38:1–38:19, Jan. 2013.

[9] J. Díaz, T. Monreal, V. Viñals, P. Ibáñez, and J. M. Llabería, "Selección de contenidos basada en reuso para caches compartidas en exclusión", in Proceedings of the XXVI Jornadas de Paralelismo, ser, 2015, pp. 433–442.

[10] J. Ahn, S. Yoo, and K. Choi, "DASCA: Dead Write Prediction Assisted STT-RAM Cache Architecture", in International Symposium on High Performance Computer Architecture (HPCA), 2014, pp. 25–36.

[11] J.C. Saez, A. Pousa, R. Rodríguez-Rodríguez, F. Castro, M. Prieto-Matías: "PMCTrack: Delivering Performance Monitoring Counter Support to the OS Scheduler". Comput. J. 60(1): 60-85 (2017).

[12] K. Nguyen, Intel's Cache Monitoring Technology Software-Visible Interfaces, https://software.intel.com/en-us/blogs/2014/12/11/intels-cache- monitoring- technology- software\- visible-interfaces, 2014.