Group Members: Hassan Farooq (hfaroo9), Spenser Fong (sfong5), Timothy Vitkin (tvitkin2)

Course project description: Partial Cache Compression With Predictive Pre-Decompression

One way to exploit the benefits of larger caches without increasing the area the cache takes up is cache compression, which compacts data stored in the cache. Cache compression can lead to higher data bandwidth and reduced overall energy consumption, at the cost of hit latency.

We propose a dynamic cache decompression technique, where cacheline entries are dynamically reallocated to either the compressed or uncompressed partitions of the cache in order to hide decompression latency.

Related Works Summary:

Compressing data in caches close to the main core is tricky due to the overhead and additional latency that occurs with the decompression necessary before data can be used [1]. Due to this, most recent work investigates compression techniques in the Last-Level Cache (LLC) of the CPU in order to increase effective cache capacity while keeping latency and physical cache size low [2]. This is especially important when using an aggressive, stride-based prefetcher that pollutes the cache, causing degradation in performance. In such cases, compression provides improved performance of up to 50% [3].

Previous work on using cache compression includes partially compressed caches, meaning that the cache has a partition for compressed data and a separate partition for uncompressed data [4, 5]. Some compression algorithms have been explored to increase compression speed, account for different data types, and reduce energy consumption [6, 7, 8]. For instance, by using chunks of four cache lines (“superblocks”) Sardashti et al. were able to reduce tag overhead in compressed caches, resulting in improved effective cache capacity without the need of significant metadata, backward pointers, or the complexity of skewed associativity [9].

Compressed LLCs that use data criticality as a consideration during compression have also been proposed, leading to 4MB compressed LLCs having performance comparable to that of an 8MB uncompressed LLC [10]. Combining compression with various replacement policies has also been explored to try and improve performance. It was found that advanced replacement policies interact poorly with compression, resulting in no noticeable improvements between compressed and uncompressed caches, but with an opportunistic cache compression mechanism, improvements of up to approximately 9% could be seen [11]. Techniques other than compression have also been tested to try and increase effective cache capacity. For instance, Huang et al. proposed a critical-words-only cache in which only words that are generally accessed before others are kept in cache resulting in a 256kB L2 cache performing just as well as a 512kB conventional L2 cache on average [12].

Due to the delay of decompression, certain techniques have been explored in order to hide the latency penalty incurred during this step [13]. Alameldeen et. al proposed a technique where an LRU and compressed data size determined whether or not data should be stored compressed or decompressed, providing an improvement of 17% for memory-intensive benchmarks. They also note that while on a typical compressed cache, a memory-intensive benchmark with a low cache miss rate incurred a performance penalty of 18%, their adaptive-compression cache only caused a 0.4% performance penalty [13]. Rea et al. proposed cache compression in addition to data prefetching in order to offset some of the latency caused by decompression specifically in L1 caches [14]. It was found that a base-delta-immediate compression combined with stride/last outcome prefetching allowed for a 1.7% average speedup compared to simply using compression without prefetching [14]. Lee et al. suggested using selective compression, parallel decompression, and the use of decompression buffers to reduce decompression overhead resulting in a 35-53% reduction in data traffic and a 20% reduction in average memory access time [15].

References:

[1] A. R. Alameldeen and R. Agarwal, "Opportunistic compression for direct-mapped dram caches," in *Proceedings of the International Symposium on Memory Systems, ser. MEMSYS '18*. New York, NY, USA: Association for Computing Machinery, 2018, p. 129–136. [Online]. Available: 10.1145/3240302.3240429.

[2] D. Chen, E. Peserico, L. Rudolph. (Nov. 2003). A Dynamically Partitionable Compressed Cache. [Online]. Available: https://dspace.mit.edu/handle/1721.1/3677.

[3] A. R. Alameldeen and D. A. Wood, "Interactions Between Compression and Prefetching in Chip Multiprocessors," in *2007 IEEE 13th International Symposium on High Performance Computer Architecture*, 2007, pp. 228-239, doi: 10.1109/HPCA.2007.346200.

[4] Data Caching with a Partially Compressed Cache, by Shanker Singh, et al. (2001, Nov. 27). US 6,324,621 B2 [Online]. Available: https://patents.google.com/patent/US6324621B2/en.

[5] D. R. Carvalho and A. Seznec, "Understanding cache compression," ACM Trans. Archit. Code Optim., vol. 18, no. 3, jun 2021. [Online]. Available: 10.1145/3457207.

[6] X. Chen, et al., "C-Pack: A High-Performance Microprocessor Cache Compression Algorithm," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 8, pp. 1196-1208, Aug. 2010, doi: 10.1109/TVLSI.2009.2020989.

[7] A. Arelakis, et al., “HyComp: a hybrid cache compression method for selection of data-type-specific compression methods,” in *MICRO-48: Proceedings of the 48th International Symposium on Microarchitecture*, 2015, pp. 38-49, doi: 10.1145/2830772.2830823.

[8] L. Villa, et al., “Dynamic zero compression for cache energy reduction,” in *MICRO 33: Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*, 2000, pp. 214-220, doi: 10.1145/360128.360150.

[9] S. Sardashti, A. Seznec, and D. A. Wood, “Yet another compressed cache: A low-cost yet effective compressed cache,” ACM Trans. Archit. Code Optim., vol. 13, no. 3, sep 2016. [Online]. Available: https://doi.org/10.1145/2976740

[10] A. Jadidi, M. Arjomand, M. T. Kandemir and C. R. Das, "Hybrid-comp: A criticality-aware compressed last-level cache," *2018 19th International Symposium on Quality Electronic Design (ISQED)*, 2018, pp. 25-30, doi: 10.1109/ISQED.2018.8357260.

[11] J. Gaur, A. R. Alameldeen, and S. Subramoney, “Base-victim compression: An opportunistic cache compression architecture,” SIGARCH Comput. Archit. News, vol. 44, no. 3, p. 317–328, jun 2016. [Online]. Available: https://doi.org/10.1145/3007787.3001171

[12] C. -C. Huang and V. Nagarajan, "Increasing cache capacity via critical-words-only cache," 2014 IEEE 32nd International Conference on Computer Design (ICCD), 2014, pp. 125-132, doi: 10.1109/ICCD.2014.6974671.

[13] A. R. Alameldeen and D. A. Wood, "Adaptive cache compression for high-performance processors," Proceedings. 31st Annual International Symposium on Computer Architecture, 2004., 2004, pp. 212-223, doi: 10.1109/ISCA.2004.1310776.

[14] S. Rea and E. Atoofian, "Mitigating Critical Path Decompression Latency in Compressed L1 Data Caches Via Prefetching," 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2018, pp. 694-701, doi: 10.1109/IPDPSW.2018.00112.

[15] J.-S. Lee, W.-K. Hong, and S.-D. Kim, “An on-chip cache compression technique to reduce decompression overhead and design complexity,” Journal of Systems Architecture, vol. 46, no. 15, pp. 1365–1382, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762100000308.