Brief Announcement: Writeback-Aware Caching*

Nathan Beckmann, Phillip B. Gibbons, Bernhard Haeupler, Charles McGuffey
{beckmann,gibbons,haeupler,cmcguffe}@cs.cmu.edu
Carnegie Mellon University

ABSTRACT

Motivated by emerging memory technologies and the increasing importance of energy and bandwidth, we study the *Writeback-Aware Caching Problem*. This problem modifies the caching problem by explicitly accounting for the cost of writing data to memory. In the offline setting with maximum writeback cost $\omega > 0$, we show that the writeback-oblivious optimal policy is only $(\omega + 1)$-competitive for writeback-aware caching, and that writeback-aware caching is NP-complete and Max-SNP hard. In the online setting, we present a deterministic online replacement policy, called *Writeback-Aware Landlord* and show that it obtains the optimal competitive ratio. Finally, we perform an experimental study on real-world traces which shows that Writeback-Aware Landlord outperforms state-of-the-art cache replacement policies when writebacks are costly.

ACM Reference Format:
Nathan Beckmann, Phillip B. Gibbons, Bernhard Haeupler, Charles McGuffey. 2019. Brief Announcement: Writeback-Aware Caching. In *31st ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '19), June 22–24, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3323165.3323169

1 INTRODUCTION

The long history of papers on caching problems [1, 3, 5–7, 18, 19, 23, 24, 26, 31–33, 37, 42, 43, 49, 50] has overlooked an increasingly important cost in real caches: the cost of *writebacks*. Whenever data that has been modified since being fetched into the cache is evicted, it must be written back to memory. In contrast, data that has not been updated since being fetched into the cache can simply be discarded from the cache on eviction. Two trends in memory systems are causing the writeback cost to become increasingly important:

Memory Bandwidth and Energy. Compared to traditional systems, modern processors have increased number of requests in-flight in the memory system. These requests provide additional parallelism, but result in many applications becoming bandwidth-limited [36]. Moreover, the end of Dennard scaling [22] has caused power consumption to become critical for a wide range of systems [20, 45]. The importance of these metrics has been underscored by a significant amount of systems research [36, 44, 47].

*Supported in part by NSF grants CCF-1533858, CCF-1618280, CCF-1814603, CCF-1527110, SHF-1815882, CCF-1725663, CCF-1750808, a Google Faculty Research Award, and a Sloan Research Fellowship.

SPAA '19, June 22–24, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6184-2/19/06.
https://doi.org/10.1145/3323165.3323169

New Memory Technologies. Several new main memory technologies that store data in the physical state of material are being developed [38]. Writing data into these memories requires more time and energy than reading data, sometimes by an order of magnitude or more [29, 34, 41]. A variety of research has been done both in the systems [2, 4, 17, 35, 40, 46, 48, 51, 52] and theory [8, 9, 11–13, 16, 17, 28, 30, 46] communities investigating this asymmetry and how it affects costs.

Related Work. Theoretical work on the original caching problem is broad and detailed. The offline version, where the request sequence is known in advance, has been solved for versions with unit size items [6, 7, 18, 37] and shown hard and approximated for versions with multiple item sizes [1, 3, 19] or similar variants [10, 14]. These offline algorithms have also been used as a baseline of comparison for online policies, which make decisions using only information from previous requests [24, 26, 43, 49, 50].

Systems research has begun to consider writebacks in caching. Initial work has been done considering partitioning [51] or tracking frequently written items [40, 48] to reduce total costs.

Unfortunately, theoretical work on writeback-aware caching is limited. Farach-Colton and Liberatore [25] showed the offline decision problem for writeback-aware caching with unit size and unit cost for both misses and writebacks is NP-complete. Belloch et al. [11] provided a writeback-aware online algorithm that is 3-competitive to offline optimal when given 3× the cache size.

Our Contributions. In this paper, we bridge the gap between real caching systems and the theoretical understanding of caches. We define and study the *Writeback-Aware Caching Problem*, a generalization of the caching problem to account for writeback costs. This problem divides accesses between writes, which modify data, and reads, which do not. Whenever an item that has been written since its previous load is evicted, a writeback occurs with a cost based on the evicted item. The goal is to minimize the sum of the costs paid for loads (misses) and the cost paid for writebacks.

Our main result is an online algorithm, called Writeback-Aware Landlord, and analysis showing that it achieves the optimal bound:

Theorem 1.1. *For the Writeback-Aware Caching Problem, Writeback-Aware Landlord with cache size k has a competitive ratio of $k/(k-h+1)$ to the optimal offline algorithm with cache size $h \leq k$.*

Although our algorithm competes with the offline optimal, computing that optimal is hard. We extend Farach-Colton and Liberatore's NP-completeness proof to show NP-completeness for any writeback costs and we show that the problem is Max-SNP hard.

We also show that FitF, the optimal writeback-oblivious algorithm, is a $(\omega + 1)$-approximation to the writeback-aware optimal.

Finally, we perform an experimental study using real-world storage traces. Writeback-Aware Landlord outperforms state-of-the-art online replacement policies when writebacks are expensive, reducing the total cost by 24% on average across these traces.

```
def WritebackAwareLandlord(Item e, bool write):
    if e is not in cache:
        while freeSpace < e.size:          # make space for the item
            minRank, victim = inf, None    # find victim
            for f in cache:
                credit = f.wbCredit + f.loadCredit
                if credit / f.size < minRank:
                    minRank = credit / f.size
                    victim = f
            evict(victim)
            for f in cache:                # decrease other items' credit
                delta = f.size * minRank
                if delta > f.wbCredit:     # decrease wb credit first
                    f.loadCredit -= (delta - f.wbCredit)
                    f.wbCredit = 0
                else:
                    f.wbCredit -= delta
        insert(e)                          # add the item to the cache
    e.loadCredit = e.loadCost              # update requested item's credit
    if write:
        e.wbCredit = e.wbCost
```

Figure 1: Writeback-Aware Landlord pseudocode.

2 AN OPTIMAL ONLINE ALGORITHM

Figure 1 gives pseudocode for Writeback-Aware Landlord. Similar to Landlord [50], Writeback-Aware Landlord assigns *credit* to each item based on its cost. Evictions remove the item whose credit to size ratio is smallest, then decrease the credit of all other cached items by their size times that ratio.

Unlike Landlord, Writeback-Aware Landlord associates *two* credits with each item, corresponding to the cost of loading the item into the cache and writing new data back to the next level of the storage hierarchy. When decreasing credit, Writeback-Aware Landlord decreases the writeback credit first.

Theorem 1.1 Proof Sketch. Consider a request trace and let O and W be the items in the cache for the optimal offline policy and Writeback-Aware Landlord. We define a potential function:

$$\Phi = (h-1)\sum_{f\in W}\left(d_l(f)+d_w(f)\right) + k\sum_{f\in O} D_l(f) + k\sum_{f\in O^*} D_w(f)$$

$$D_l(f) = t_l(f) - d_l(f) \quad \text{and} \quad D_w(f) = t_w(f) - d_w(f)$$

where O^* is the set of items in O that are dirty, d and t refer to credit and cost, and the subscripts l and w refer to load and writeback, respectively. We show the following:

(1) Φ is zero at the beginning of the trace.
(2) Φ is always non-negative.
(3) Each cost c paid by Writeback-Aware Landlord can be charged to a unique decrease in Φ of at least $(k-h+1)c$.
(4) Φ can only ever increase by an amount kc when the optimal algorithm pays a cost c.

These facts show that the cost ratio of Writeback-Aware Landlord to optimal cannot exceed $k/(k-h+1)$. □

This matches the lower bound proven by Sleator and Tarjan [43], showing that it is optimal in terms of worst-case analysis.

3 OFFLINE COMPLEXITY RESULTS

NP-Completeness. We extend the proof of [33] to show that for *any* writeback cost $\omega > 0$, the offline problem is NP-Complete.
Max-SNP-Hardness. We show that Offline Writeback-Aware Caching is Max-SNP-hard using a reduction from bounded 3-dimensional matching. Our reduction shows that any approximation algorithm for Offline Writeback-Aware Caching can be used to generate an approximation algorithm for bounded 3-dimensional matching.
Furthest-in-the-Future. The FitF algorithm solves the writeback-oblivious caching problem when all items have unit size and cost. We show how algorithms like this are non-optimal with writebacks.

Theorem 3.1. *FitF is an $\omega+1$ approximation for Writeback-Aware Caching with unit sizes and load costs. This bound is tight.*

4 EXPERIMENTAL EVALUATION

To show that the theory behind Writeback-Aware Landlord holds up well in practice, we evaluated Writeback-Aware Landlord against the LRU and Greedy Dual Size (GDS) [15] policies on five storage traces from Microsoft Research (MSR) [39]. These traces represent many commonly seen workload behaviors. LRU is the simplest policy commonly used in practice. It treats all items equally, ignoring the size and cost of items. GDS is an efficient implementation of the original (writeback-oblivious) Landlord algorithm, which considers item cost and size when making decisions.

Fair comparisons against prior writeback-aware policies [40, 47, 51] are not possible because these policies assume items have fixed size, whereas in our traces item sizes vary. This would cause these prior policies to perform poorly for reasons unrelated to writebacks.
Metrics. Since MSR trace requests do not specify cost, we consider two accounting methods. In the fault model, each item has unit load cost and writeback cost ω. This represents a system where communication cost is largely independent of size, i.e., latency trumps bandwidth. In the bit model these costs are multiplied by the size of the requested item. This models a system where communication is charged on a per byte basis, such as bandwidth-constrained memory. We set $\omega = 10$, which approximates the read-write asymmetry of flash memory [27] and emerging technologies [21].
Implementation. Writeback-Aware Landlord as described in Figure 1 requires work linear in the number of cached items per eviction. We implement an equivalent version based on GDS' improvements to Landlord that requires only logarithmic work per eviction.
Results. Writeback-Aware Landlord outperforms GDS and LRU by a fairly uniform margin across four of the five MSR traces for cache sizes from 16 to 512 GB. One trace (*src1_0*) has access patterns that do not benefit from Writeback-Aware Landlord.

By geometric mean across traces and cache sizes, Writeback-Aware Landlord's miss cost is the same as GDS (up to three significant figures), while reducing writeback cost by 36%. The result is that the Writeback-Aware Landlord's total cost ranges from 12–100% of GDS and 8.7–110% of LRU, with a geometric mean cost of 76% of GDS and 59% of LRU.

We see similar results for different writeback costs, performance metrics, and additional heuristics.

ACKNOWLEDGMENTS

We thank David Wajc for his contributions to the reduction from 3-dimensional matching.

REFERENCES

[1] Susanne Albers, Sanjeev Arora, and Sanjeev Khanna. 1999. Page replacement for general caching problems. In *SODA*, Vol. 99. Citeseer, 31–40.

[2] Joy Arulraj and Andrew Pavlo. 2017. How to build a non-volatile memory database management system. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 1753–1758.

[3] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. 2001. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM (JACM)* 48, 5 (2001), 1069–1090.

[4] Daniel Bausch, Ilia Petrov, and Alejandro Buchmann. 2012. Making cost-based query optimization asymmetry-aware. In *Proceedings of the Eighth International Workshop on Data Management on New Hardware*. ACM, 24–32.

[5] Nathan Beckmann and Daniel Sanchez. 2017. Maximizing cache performance under uncertainty. In *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*. IEEE, 109–120.

[6] Laszlo A. Belady. 1966. A study of replacement algorithms for a virtual-storage computer. *IBM Systems journal* 5, 2 (1966), 78–101.

[7] Laszlo A. Belady and Frank P. Palermo. 1974. On-line measurement of paging behavior by the multivalued MIN algorithm. *IBM Journal of Research and Development* 18, 1 (1974), 2–19.

[8] Naama Ben-David, Guy E. Blelloch, Jeremy T. Fineman, Phillip B. Gibbons, Yan Gu, Charles McGuffey, and Julian Shun. 2016. Parallel algorithms for asymmetric read-write costs. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. ACM, 145–156.

[9] Naama Ben-David, Guy E. Blelloch, Jeremy T. Fineman, Phillip B. Gibbons, Yan Gu, Charles McGuffey, and Julian Shun. 2018. Implicit Decomposition for Write-Efficient Connnectivity Algorithms. In *International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 711–722.

[10] Daniel S. Berger, Nathan Beckmann, and Mor Harchol-Balter. 2018. Practical Bounds on Optimal Caching with Variable Object Sizes. *Proc. ACM Meas. Anal. Comput. Syst. (SIGMETRICS'18)* (2018). https://doi.org/10.1145/3224427

[11] Guy E. Blelloch, Jeremy T. Fineman, Phillip B. Gibbons, Yan Gu, and Julian Shun. 2015. Sorting with asymmetric read and write costs. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. ACM, 1–12.

[12] Guy E. Blelloch, Jeremy T. Fineman, Phillip B. Gibbons, Yan Gu, and Julian Shun. 2016. Efficient algorithms with asymmetric read and write costs. In *European Symposium on Algorithms*.

[13] Guy E. Blelloch, Yan Gu, Julian Shun, and Yihan Sun. 2018. Parallel write-efficient algorithms and data structures for computational geometry. In *Proceedings of the 30th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. ACM, 235–246.

[14] Mark Brehob, Stephen Wagner, Eric Torng, and Richard Enbody. 2004. Optimal replacement is NP-hard for nonstandard caches. *IEEE Transactions on computers* 53, 1 (2004), 73–76.

[15] Pei Cao and Sandy Irani. 1997. Cost-aware www proxy caching algorithms.. In *Usenix symposium on internet technologies and systems*, Vol. 12. 193–206.

[16] Erin Carson, James Demmel, Laura Grigori, Nicholas Knight, Penporn Koanantakool, Oded Schwartz, and Harsha Vardhan Simhadri. 2016. Write-avoiding algorithms. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 648–658.

[17] Shimin Chen, Phillip B. Gibbons, and Suman Nath. 2011. Rethinking Database Algorithms for Phase Change Memory. In *Proc. Conference on Innovative Data Systems Research (CIDR)*.

[18] Marek Chrobak, Howard J. Karloff, T. H. Payne, and Sundar Vishwanathan. 1991. New Results on Server Problems. In *SIAM Journal on Discrete Mathematics*. 172–181.

[19] Marek Chrobak, Gerhard J. Woeginger, Kazuhisa Makino, and Haifeng Xu. 2012. Caching is hard-even in the fault model. *Algorithmica* 63, 4 (2012), 781–794.

[20] Alexei Colin and Brandon Lucia. 2018. Termination checking and task decomposition for task-based intermittent programs. In *Proceedings of the 27th International Conference on Compiler Construction*. ACM, 116–127.

[21] Intel Corporation. 2018. Optane SSD DC P4800X Series. Retrieved online on 11 Jan 2019 at https://ark.intel.com/products/97161/Intel-Optane-SSD-DC-P4800X-Series-375GB-2-5in-PCIe-x4-3D-XPoint-.

[22] Robert H. Dennard, Fritz H. Gaensslen, V. Leo Rideout, Ernest Bassous, and Andre R. LeBlanc. 1974. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits* 9, 5 (1974), 256–268.

[23] Nam Duong, Dali Zhao, Taesu Kim, Rosario Cammarota, Mateo Valero, and Alexander V. Veidenbaum. 2012. Improving cache management policies using dynamic reuse distances. In *Microarchitecture (MICRO), 2012 45th Annual IEEE/ACM International Symposium on*. IEEE, 389–400.

[24] Guy Even, Moti Medina, and Dror Rawitz. 2018. Online generalized caching with varying weights and costs. In *Proceedings of the 30th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. ACM, 205–212.

[25] Martin Farach-Colton and Vincenzo Liberatore. 2000. On local register allocation. *Journal of Algorithms* 37, 1 (2000), 37–65.

[26] Amos Fiat, Richard M. Karp, Michael Luby, Lyle A McGeoch, Daniel D. Sleator, and Neal E. Young. 1991. Competitive paging algorithms. *Journal of Algorithms* 12, 4 (1991), 685–699.

[27] Laura M. Grupp, Adrian M. Caulfield, Joel Coburn, Steven Swanson, Eitan Yaakobi, Paul H. Siegel, and Jack K. Wolf. 2009. Characterizing flash memory: anomalies, observations, and applications. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*. IEEE, 24–33.

[28] Yan Gu, Yihan Sun, and Guy E. Blelloch. 2018. Algorithmic building blocks for asymmetric memories. In *European Symposium on Algorithms*. 44:1–44:15.

[29] IBM 2014. www.slideshare.net/IBMZRL/theseus-pss-nvmw2014.

[30] Riko Jacob and Nodari Sitchinava. 2017. Lower bounds in the asymmetric external memory model. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. ACM, 247–254.

[31] Akanksha Jain and Calvin Lin. 2016. Back to the future: leveraging Belady's algorithm for improved cache replacement. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*. IEEE, 78–89.

[32] Aamer Jaleel, Kevin B. Theobald, Simon C. Steely Jr, and Joel Emer. 2010. High performance cache replacement using re-reference interval prediction (RRIP). In *ACM SIGARCH Computer Architecture News*, Vol. 38. ACM, 60–71.

[33] Georgios Keramidas, Pavlos Petoumenos, and Stefanos Kaxiras. 2007. Cache replacement based on reuse-distance prediction. In *Computer Design, 2007. ICCD 2007. 25th International Conference on*. IEEE, 245–250.

[34] Hyojun Kim, Sangeetha Seshadri, Clement L. Dickey, and Lawrence Chiu. 2014. Evaluating phase change memory for enterprise storage systems: A study of caching and tiering approaches. *ACM Transactions on Storage (TOS)* 10, 4 (2014), 15.

[35] Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. 2009. Architecting phase change memory as a scalable dram alternative. In *ACM SIGARCH Computer Architecture News*, Vol. 37. ACM, 2–13.

[36] Chang Joo Lee, Veynu Narasiman, Eiman Ebrahimi, Onur Mutlu, and Yale N. Patt. 2010. *DRAM-aware last-level cache writeback: Reducing write-caused interference in memory systems.* Technical Report. U.T. Austin.

[37] Richard L. Mattson, Jan Gecsei, Donald R. Slutz, and Irving L. Traiger. 1970. Evaluation techniques for storage hierarchies. *IBM Systems journal* 9, 2 (1970), 78–117.

[38] Jagan Singh Meena, Simon Min Sze, Umesh Chand, and Tseung-Yuen Tseng. 2014. Overview of emerging nonvolatile memory technologies. *Nanoscale research letters* 9, 1 (2014), 526.

[39] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. 2008. Write off-loading: Practical power management for enterprise storage. *ACM Transactions on Storage (TOS)* 4, 3 (2008), 10.

[40] Hanfeng Qin and Hai Jin. 2017. Warstack: Improving LLC Replacement for NVM with a Writeback-Aware Reuse Stack. In *Parallel, Distributed and Network-based Processing (PDP), 2017 25th Euromicro International Conference on*. IEEE, 233–236.

[41] Moinuddin K. Qureshi, Sudhanva Gurumurthi, and Bipin Rajendran. 2011. Phase change memory: From devices to systems. *Synthesis Lectures on Computer Architecture* 6, 4 (2011), 1–134.

[42] Moinuddin K. Qureshi, Aamer Jaleel, Yale N. Patt, Simon C. Steely, and Joel Emer. 2007. Adaptive insertion policies for high performance caching. In *ACM SIGARCH Computer Architecture News*, Vol. 35. ACM, 381–391.

[43] Daniel D. Sleator and Robert E. Tarjan. 1985. Amortized efficiency of list update and paging rules. *Commun. ACM* 28, 2 (1985), 202–208.

[44] Jeffrey Stuecheli, Dimitris Kaseridis, David Daly, Hillery C. Hunter, and Lizy K. John. 2010. The virtual write queue: Coordinating DRAM and last-level cache policies. *ACM SIGARCH Computer Architecture News* 38, 3 (2010), 72–82.

[45] Qinghui Tang, Sandeep Kumar S. Gupta, and Georgios Varsamopoulos. 2008. Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach. *IEEE Transactions on Parallel and Distributed Systems* 19, 11 (2008), 1458–1472.

[46] Stratis D. Viglas. 2014. Write-limited sorts and joins for persistent memory. *Proceedings of the VLDB Endowment* 7, 5 (2014), 413–424.

[47] Zhe Wang, Samira M. Khan, and Daniel A. Jiménez. 2012. Improving writeback efficiency with decoupled last-write prediction. In *ACM SIGARCH Computer Architecture News*, Vol. 40. IEEE Computer Society, 309–320.

[48] Zhe Wang, Shuchang Shan, Ting Cao, Junli Gu, Yi Xu, Shuai Mu, Yuan Xie, and Daniel A. Jiménez. 2013. WADE: Writeback-aware dynamic cache management for NVM-based main memory system. *ACM Transactions on Architecture and Code Optimization (TACO)* 10, 4 (2013), 51.

[49] Neal Young. 1994. The k-server dual and loose competitiveness for paging. *Algorithmica* 11, 6 (1994), 525–541.

[50] Neal E. Young. 2002. On-line file caching. *Algorithmica* 33, 3 (2002), 371–383.

[51] Miao Zhou, Yu Du, Bruce Childers, Rami Melhem, and Daniel Mossé. 2012. Writeback-aware partitioning and replacement for last-level caches in phase change main memory systems. *ACM Transactions on Architecture and Code Optimization (TACO)* 8, 4 (2012), 53.

[52] Omer Zilberberg, Shlomo Weiss, and Sivan Toledo. 2013. Phase-change memory: An architectural perspective. *ACM Computing Surveys (CSUR)* 45, 3 (2013), 29.