A Dual Grain Hit-Miss Detector for Large Die-Stacked DRAM Caches

Michel El-Nacouzi, Islam Atta, Myrto Papadopoulou, Jason Zebchuk, Natalie Enright Jerger and Andreas Moshovos
Electrical and Computer Engineering
University of Toronto
{elnacouz, iatta, myrto, zebchuk, enright, moshovos}@eecg.toronto.edu

***Abstract*—Die-Stacked DRAM caches offer the promise of improved performance and reduced energy by capturing a larger fraction of an application's working set than on-die SRAM caches. However, given that their latency is only 50% lower than that of main memory, DRAM caches considerably increase latency for misses. They also incur a significant energy overhead for remote lookups in snoop-based multi-socket systems. Ideally, it would be possible to detect in advance that a request will miss in the DRAM cache and thus selectively bypass it. This work proposes a dual grain filter which successfully predicts whether an access is a hit or a miss in most cases. Experimental results with commercial and scientific workloads show that a 158KB dual-grain filter can correctly predict data block residency for 85% of all accesses to a 256MB DRAM cache. As a result, off-die latency with our filter is nearly identical to that possible with an impractical, perfect filter.**

I. Introduction

Die-stacked DRAM Caches (DRAMC) are projected to have capacities at least an order of magnitude larger than existing on-die SRAM caches. These large capacities allow DRAMCs to capture a large portion of an application's memory footprint and thus can significantly improve system performance. However, using a DRAMC is not free of challenges. The increased capacity compared to last-level on-die caches comes at the cost of higher access latency and energy per access. Therefore using a DRAMC in the same way as a conventional on-die cache may result in a considerable latency and energy increase for some memory accesses.

As a DRAMC is faster than off-chip main memory, it can reduce latency for those accesses that hit in the cache. However, the latency of misses would increase compared to a system without a DRAMC. The smaller latency difference between a DRAMC and main memory, compared to a conventional SRAM cache, makes performance more sensitive to DRAMC hit rate. Ideally, to reap the full benefits of a DRAM cache, one would like to know in advance if an access would miss in the DRAMC, so that it can bypass it and directly access the off-chip memory. Selectively bypassing a DRAMC for those accesses that would miss in it, would not only improve performance but also reduce energy by avoiding unnecessary DRAMC accesses.

Besides improving performance for misses from the local cores, bypassing the DRAMC can also benefit multi-socket snoop-based systems. Most remote snoop accesses rarely find their data on remote nodes, and many techniques have been proposed to filter the corresponding snoops (e.g., [8]). A DRAMC miss filter would avoid such unnecessary lookups reducing the latency and energy for remote snoop requests.

This work aims at developing an energy and area efficient miss detection filter for DRAMCs, reducing the latency and power overhead of unnecessary DRAMC tag lookups. The proposed Dual-Grain Filter (DUAL-GRAIN$_F$) comprises two underlying filters: the Fine-Grain Filter (FINE$_F$) which keeps track of blocks in the cache at the block level, and the Coarse-Grain Filter (COARSE$_F$) which does so at the page level. DUAL-GRAIN$_F$ has been designed to accurately predict most hits and misses, using space-efficient structures that exploit common program behavior for applications with relatively large memory footprints.

This work demonstrates the benefits of DUAL-GRAIN$_F$ when local misses bypass the DRAMC. Experimental results with commercial and scientific applications demonstrate that a 158KB DUAL-GRAIN$_F$ successfully predicts 85% of all accesses resulting in an off-die latency that is close to that possible with an ideal predictor.

II. Motivation

Selectively bypassing the cache for misses results in an ideal situation where we get the energy and performance benefits of hits with none of the overheads of misses. This work proposes DUAL-GRAIN$_F$ which uses a dual grain approach to identify hits and misses for large die-stacked caches. DUAL-GRAIN$_F$ judiciously uses storage and exploits typical program behavior to better capture DRAMC misses.

To explain the reasoning behind the DUAL-GRAIN$_F$ design let us first consider an application that streams through an array experiencing compulsory misses. DUAL-GRAIN$_F$ will try to detect the first miss into a page by using COARSE$_F$ to track which pages currently have blocks cached in DRAMC. For most non-cached pages, the expectation is that COARSE$_F$ will report a miss when the first request arrives. Once this *first miss* is detected, FINE$_F$ starts tracking this page's blocks as they are allocated. Because programs exhibit spatial locality, we expect that, at any given point in time, there will be few pages per core that FINE$_F$ needs to track in order to detect most compulsory misses.

978-3-9815370-0-0/DATE13/©2013 EDAA

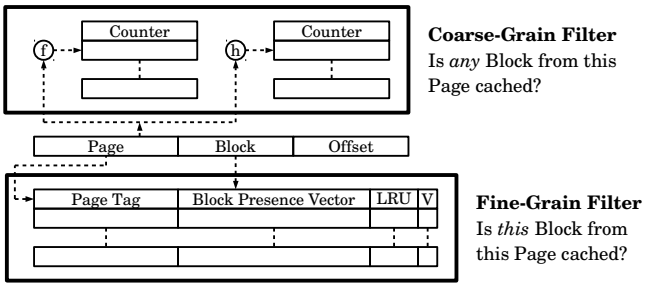

Fig. 1. Structure of Dual-Grain Filter

Now let us consider the case where the array does not fit in the cache. With an LRU replacement policy subsequent passes of the array will result in capacity misses. By the time a page is revisited, all of its blocks will have been evicted and thus the COARSE$_F$ will successfully detect the first miss to each page. This again allows a small FINE$_F$ to track all subsequent capacity misses to the same page. With a non-LRU policy, some part of the array might remain cache resident. In this case, COARSE$_F$ should detect hits to pages that are cache resident, while misses to non-resident pages are still detected the same as compulsory misses.

In general, DUAL-GRAIN$_F$ was designed with the following expectations: (a) Applications tend to exhibit spatial locality so that a first access to a page will soon be followed by accesses to other blocks in the same page. This suggests that by tracking a few pages at any point we should be able to detect hits and misses for most accesses. (b) At any given point in time there will be a few pages that experience misses and in most cases these will come soon after observing a first miss. (c) If a page experiences a hit, then most likely, the rest of the accesses to the page will be hits as well. (d) Very few pages will experience a mix of hits and misses once cache resident. Section IV demonstrates that most scientific and commercial applications meet these expectations.

III. DUAL-GRAIN$_F$ DESIGN

As Fig. 1 shows, DUAL-GRAIN$_F$ comprises FINE$_F$ which tracks all blocks in selected pages, and COARSE$_F$ which tracks a superset of pages that have cached blocks in DRAMC. (A page is 4KB and a block is 64B.) These two structures are implemented using SRAM arrays on the processor die.

A. Coarse-Grain Filter

As indicated in Fig. 1, COARSE$_F$ is a counting Bloom filter that tracks data residency in DRAMC at a coarse-grain level. In this work, COARSE$_F$ tracks data at the page level. An incoming request uses its page number and the hashing functions f and h to index into two counting Bloom tables. As blocks are allocated and de-allocated, the corresponding COARSE$_F$ counters are incremented or decremented respectively. A particular page, when hashed into different tables, has no blocks cached in DRAMC if *any* of the table counters are zero. If no aliasing exists, COARSE$_F$ would perfectly capture first misses to pages and trigger the allocation of corresponding entries in FINE$_F$. In practice, however, aliasing could cause a page with no cached blocks to have non-zero counters.

In the worst case scenario, all blocks can alias to a single counter, requiring 22-bit counters for a 256MB DRAMC. In practice, however, each counter is only used by the blocks from a few pages. To handle this common case, COARSE$_F$ uses 8-bit counters. When a counter overflows, an additional *grouped* bit is set, and the counter is combined with three adjacent counters to form a single 32-bit counter. When the combined counter reaches zero, the grouped bit is reset and the combined counter becomes four independent counters again.

B. Fine-Grain Filter

FINE$_F$ is a set associative structure that tracks a few recently accessed pages at the block level. As Fig. 1 shows, a FINE$_F$ entry contains a 36-bit tag identifying the page, a 64-bit *presence* vector indicating which blocks of this page are present in DRAMC, LRU bits (or any other replacement information), and a valid bit.

There are three possible outcomes on a FINE$_F$ probe: (1) The block is cached (HIT), (2) the block is not cached (MISS), or (3) there is no information about that block (X). This last case occurs when no valid entry is found for the page in question. Due to its limited size, FINE$_F$ tracks only a subset of pages. Once it becomes full, allocating new entries requires evicting and discarding information about an existing page. The evicted page remains implicitly in the X state until all of its blocks have been evicted and COARSE$_F$ indicates the page is not present in the cache.

C. Dual-Grain Filter

DUAL-GRAIN$_F$ is a combination of FINE$_F$ and COARSE$_F$ and operates as follows: First we access COARSE$_F$ by hashing the page number into two Bloom tables. If any of the counters is 0, this means that there is a Miss. Otherwise, we have to access FINE$_F$ to check the outcome. If no entry is found, the outcome is unknown (X) and accordingly, we should check DRAMC first, and if the block is not cached, then we obtain the data from off-chip DRAM.

In designing DUAL-GRAIN$_F$, we observe that some information is duplicated in both COARSE$_F$ and FINE$_F$. We can exploit this to reduce aliasing in COARSE$_F$. Specifically, as long as an entry for a page exists in FINE$_F$ we do not modify the COARSE$_F$ counters upon allocation/eviction of blocks from that page. On an eviction of a FINE$_F$ entry, these counters are incremented by the number of blocks indicated by that entry. Since the two filters are now exclusive we have to wait for both before deciding to bypass the DRAMC. This configuration, exclusive DUAL-GRAIN$_F$, provides better accuracy for a large-sized FINE$_F$, but for a small FINE$_F$ it has almost no effect. In the evaluation section, we will consistently be using the exclusive DUAL-GRAIN$_F$ version.

IV. EVALUATION

The evaluation section is organized as follows: Section IV-A presents the evaluation methodology. Section IV-B shows how DUAL-GRAIN$_F$ coverage and accuracy change with the size of FINE$_F$ and COARSE$_F$. Section IV-C shows that a 158KB DUAL-GRAIN$_F$ results in DRAMC performance that is close to that possible with a perfect hit/miss predictor.

TABLE I. SYSTEM PARAMETERS

Processor	16-cores, 3.2GHz
Private L1i, L1d	32KB, 64B blocks, 4-way
Private L2	256KB, 64B blocks, 8-way
Shared L3	16MB, 6B blocks, 16-way
Die-stacked DRAMC L4	256MB w/ tags, 29-way, 64B blocks, 1.6GHz DDR, t_{CAS} 9, t_{ACT} 9, t_{BUS} 4
Off-chip DRAM	800MHz DDR, t_{CAS} 9, t_{ACT} 9, t_{BUS} 4, 15ns offchip delay

TABLE II. WORKLOADS (MEMORY FOOTPRINT)

SPLASH-2	Em3d (131MB), Ocean (209MB), Sparse (203MB)
Parsec (Native)	Blackscholes (271MB), Canneal (307MB), X264 (70MB), Ferret (90MB), Swaptions (1.5MB)
Decision Support Sys.	TPCH-qry1 (926MB), TPCH-qry2 (889MB),
Online Trans. Proc.	DB2 (252MB), Oracle (295MB)
SpecWEB 99	Apache (617MB), Zeus (183MB)
Java Server	SpecJBB (370MB)

A. Methodology

We model a 16-core SPARCv9 chip-multiprocessor as described in Table I. We use Flexus simulator [3], which builds on WindRiver Simics, a full system simulator. The tags are stored directly in the DRAM array with the data [5]. We assume a 2KB DRAM row in the DRAMC with 32×64B blocks per row. Using 6B tags, the row is partitioned into 29 blocks for data, and 3 blocks for tags. We simulated 1 billion instructions per core for each of the benchmarks listed in Table II, except for Sparse and TPCH_qry2 which finished at 485 and 386 million instructions respectively. The memory footprints shown in parentheses in Table II indicate that these workloads make use of the 256MB DRAMC, and experimental results not included here indicate that a larger DRAMC does not provide significant benefits. Thus, we chose a 256MB DRAMC for our evaluation.

B. Filter Accuracy

This section demonstrates how accurate DUAL-GRAIN$_F$ is at detecting DRAMC misses. Inaccuracy occurs only when a request hits in COARSE$_F$ and fails to find a FINE$_F$ entry. In this scenario (the X case), DUAL-GRAIN$_F$ provides no information about whether the block is in the DRAMC or not. Fig. 2 shows the accuracy of different DUAL-GRAIN$_F$ configurations, where the accuracy is measured as the fraction of requests that do not result in the X case. Increasing the COARSE$_F$ beyond 132KB has little effect on overall accuracy. The accuracy is much more dependant on FINE$_F$ size, with significant improvements as the size increases up to 1664KB. At that size, DUAL-GRAIN$_F$ can achieve an accuracy of 98%, but this requires significant storage overhead. However, the accuracy shown in Fig. 2 pessimistically assumes that all X cases are inaccurate. In practice, the DRAMC is accessed for all X cases, and some of these accesses will hit in the DRAMC, indicating an accurate prediction from the DUAL-GRAIN$_F$.

Fig. 3 shows a breakdown of the predictions made by a DUAL-GRAIN$_F$ with a 132KB COARSE$_F$ and a small, 26KB 8-way set associative FINE$_F$. As expected X outcomes now dominate; however, with the exception of SpecJBB and Blackscholes, most of these X outcomes are for DRAMC hits.

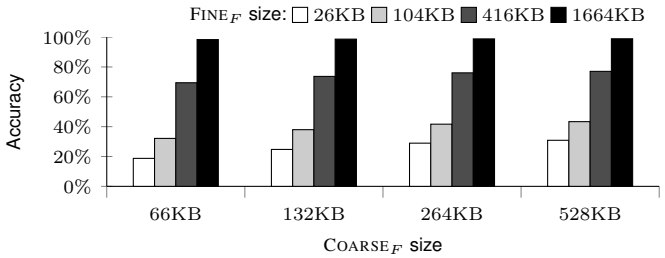

Fig. 2. Accuracy of DUAL-GRAIN$_F$ as a function of COARSE$_F$ and FINE$_F$ size. All COARSE$_F$ configurations use two equally sized tables, and all FINE$_F$ configurations are 8-way set associative.

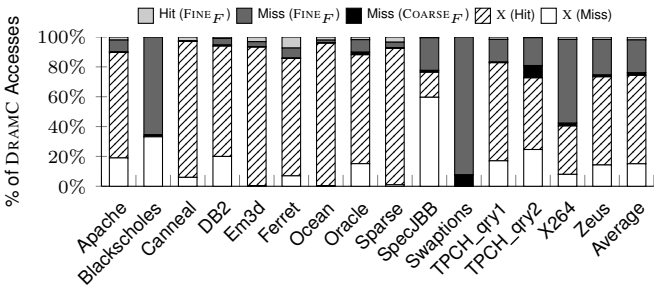

Fig. 3. Detailed breakdown of predictions - 158KB DUAL-GRAIN$_F$

TABLE III. PRACTICAL FILTER CONFIGURATIONS

Filter	Configuration	Total Storage
DUAL-GRAIN$_F$		**158KB**
COARSE$_F$	2 tables, each with 64K 8-bit counters, one grouped bit for each 4 counters	132KB
FINE$_F$	8-way set associative, 256 sets, 104 bits/entry	26KB
MISSMAP	12-way set associative, 1K sets, 104 bits/entry	**156KB**

When these hits are included in the accuracy calculation, the average accuracy for this design is 85%, making it a much more promising design point. This behavior confirms the expectations from Section II that if one block in a page hits in the DRAMC, other blocks from the same page will also hit.

C. Benefits for Off-Die Accesses

This section shows that DUAL-GRAIN$_F$, when compared to a similarly sized MISSMAP, can avoid a reduction in DRAMC hit rate and reduce average latency for off-die accesses. We use similarly sized MISSMAP and DUAL-GRAIN$_F$ configurations as described in Table III.

Fig. 4 shows the DRAMC miss rate for both DUAL-GRAIN$_F$ and MISSMAP. When a MISSMAP entry is evicted, all DRAMC blocks tracked by that entry are evicted from the DRAMC to keep the MISSMAP precise. As Fig. 4 demonstrates, these evictions significantly increase the DRAMC miss rate for most workloads when using a small MISSMAP design. There are a few exceptions, such as *swaptions*, which has a very small working set, but on average MISSMAP has a DRAMC miss rate that is 117% higher than DUAL-GRAIN$_F$.

Fig. 5 shows how off-die (post L3) memory latency varies for three configurations. The first uses a perfect predictor,

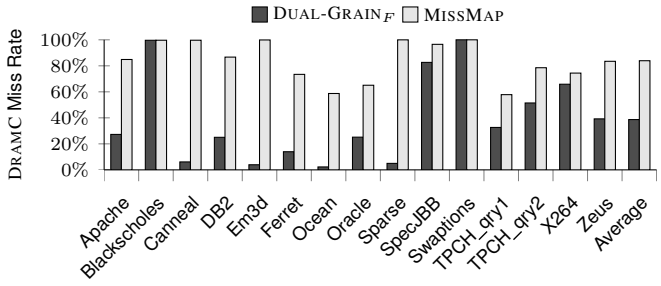

Fig. 4. Effect of forced evictions on DRAMC miss rate.

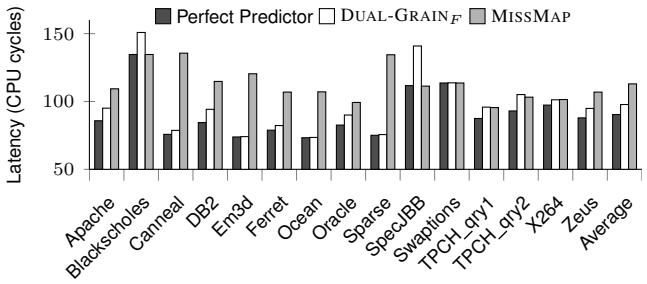

Fig. 5. Post-L3 latency.

the second is a 158KB DUAL-GRAIN$_F$, and the third is a MISSMAP that uses 156KB. As estimated by CACTI 6.5 [9], DUAL-GRAIN$_F$ and MISSMAP require 4 and 5 cycles respectively in 32nm technology, but we do not include this in Fig. 5. The latencies in Fig. 5 are estimated using fixed latency numbers derived from those in Table III. As expected, DUAL-GRAIN$_F$ reduces off-die latency compared to a similarly sized MISSMAP. Moreover, the off-die latency with DUAL-GRAIN$_F$ is only 8% higher, on average, compared to a perfect predictor.

V. RELATED WORK

The emergence of stacked-DRAM as a last level cache brings new challenges, such as long latency tag lookups and increased power consumption. MISSMAP [5] and CHOP [4] are two recent proposals that alleviate some of these penalties. MISSMAP is a precise data structure which provides cache residency information for a given address. Although the authors mention the option of an imprecise MISSMAP with the use of Bloom filters, they do not explore the intricacies of such a design. CHOP proposes a set of filter-based caching techniques for DRAM at the page level. Our work, however, focuses on caching at the block level.

The most recent studies on DRAM caches try to balance the traffic between DRAMC and main memory [11], or try to optimize for latency by using a direct-mapped DRAMC [10]. Both use a small predictor to further optimize their design, but these predictors cannot be used for snoop filtering or other optimizations that rely on miss detection.

Cache filtering techniques have also been proposed in the past for SRAM caches. Memik et al. proposed a set of techniques that predict misses for different caches, bypassing the respective cache-levels [6], and Zebchuk and Moshovos used dual-grain tracking to avoid tag lookups for SRAM caches [13]. Bloom filters have been proposed for filtering snoops [8], [7], and for coarse-grain tracking of blocks in last-level caches [12]. Ghosh et al. proposed using a segmented counting Bloom filter to avoid accessing all ways in a tag-lookup of a large set-associative cache, thus reducing dynamic energy [2]. In our work, we use Bloom filters along with a combination of coarse-grain and fine-grain address tracking to predict cache residency information for die-stacked DRAM. The large capacity of the DRAM caches requires a re-evaluation of prior-art as the reach of a Bloom filter is now an order of magnitude larger.

VI. CONCLUSION

We have demonstrated that a dual-grain approach can be used to successfully predict most misses prior to accessing a large stacked die cache for both commercial and scientific workloads. DUAL-GRAIN$_F$ avoids accessing the DRAMC for misses, thus improving performance and energy. We demonstrate that a DUAL-GRAIN$_F$ design requiring approximately 158KB of storage achieves an accuracy of 85% in predicting DRAMC misses, and results in an average off-die latency that is almost as low as with a perfect miss detector.

REFERENCES

[1] B.H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. In *Commun. ACM* 13, 7, July 1970.

[2] M. Ghosh, E. Ozer, S. Ford, S. Biles, and H. S. Lee. 2009. Way guard: a segmented counting bloom filter approach to reducing energy for set-associative caches. In *Proc. of the 14th ACM/IEEE Int'l Symp. on Low power electronics and design*, 2009.

[3] N. Hardavellas, et al. SimFlex: a Fast, Accurate, Flexible Full-System Simulation Framework for Performance Evaluation of Server Architecture. In *SIGMETRICS Perform. Eval. Rev.* 31, 4, March 2004.

[4] X. Jiang, et al., CHOP: Integrating DRAM Caches for CMP Server Platforms, In *IEEE Micro Top Picks*, 2011.

[5] G.H. Loh and M.D. Hill. Efficiently Enabling Conventional Block Sizes for Very Large Die-Stacked DRAM Caches. In *Proc. of 44th Int'l Symp. on Microarchitecture*, 2011.

[6] G. Memik, G. Reinman, and W.H. Mangione-Smith. Just Say No: Benefits of Early Cache Miss Determination. In *Proc. of 9th Int'l Symp. on High-Perf. Comp. Architecture*, 2003.

[7] A. Moshovos. RegionScout: Exploiting Coarse Grain Sharing in Snoop-Based Coherence. In *Proc. of 32nd Int'l Symp. on Comp. Architecture*, 2005.

[8] A. Moshovos, G. Memik, B. Falsafi, and A. Choudhary. JETTY: Filtering Snoops for Reduced Energy Consumption in SMP Servers. In *Proc. of 7th Int'l Symp. on High-Perf. Comp. Architecture*, 2001.

[9] N. Muralimanohar, R. Balasubramonian, and N.P. Jouppi. CACTI 6.0: A Tool to Model Large Caches. HP Laboratories, Tech. Rep. HPL-2009-85, 2009.

[10] M.K. Qureshi and G.H. Loh. Fundamental Latency Trade-offs in Architecting DRAM Caches. In *Proc. of 45th Int'l Symp. on Microarchitecture*, 2012.

[11] J. Sim, et al. A Mostly-Clean DRAM Cache for Effective Hit Speculation and Self-Balancing Dispatch. In *Proc. of 45th Int'l Symp. on Microarchitecture*, 2012.

[12] J. Zebchuk, E. Safi, and A. Moshovos. A Framework for Coarse-Grain Optimizations in the On-Chip Memory Hierarchy. In *Proc. of 40th Int'l Symp. on Microarchitecture*, 2007.

[13] J. Zebchuk, and A. Moshovos. RegionTracker: Using Dual-Grain Tracking for Energy Efficient Cache Lookup. Presented at the Workshop on Complexity Effective Design, co-located with the *Int'l Symp. on Computer Architecture*, 2006.