OAP: An Obstruction-Aware Cache Management Policy for STT-RAM Last-Level Caches

Jue Wang, Xiangyu Dong†, Yuan Xie
Department of Computer Science and Engineering, Pennsylvania State University
Qualcomm Technology, Inc.†
{jzw175,yuanxie}@cse.psu.edu xiangyud@qti.qualcomm.com†

Abstract—Deeper cache hierarchy and larger cache capacity are observed in modern multi-core systems, and a large L3 cache is becoming a default component recently. To achieve small chip size and low energy requirements of L3 caches, we are likely to use the Spin-Torque Transfer RAM (STT-RAM) technology instead of the conventional SRAM or embedded DRAM. However, as the last-level cache (LLC), L3 cache is usually associated with single-port bitcell designs, which greatly amplify the shortcoming of STT-RAM in write speed. To cope with this, we advocate OAP, an obstruction-aware cache management policy. OAP first adds monitors to the conventional cache architecture to periodically detect LLC-obstruction processes. Then, based on the detection result, OAP controls L3 cache accesses from different processes. After adopting OAP, the performance of a projected 4-core system with an 8MB STT-RAM L3 cache is improved by 14% on average and up to 42%. In addition, OAP further reduces the energy consumption of L3 cache by 64%.†

I. INTRODUCTION

As the number of processor cores keeps increasing, more design pressure has been put on memory subsystem performance improvement. As a result, deeper cache hierarchy and larger cache capacity can be observed in contemporary microprocessor designs. For example, Intel i7-3930K processor is equipped with a 12MB SRAM L3 cache, and IBM POWER7 processor has an unprecedented 32MB embedded DRAM (eDRAM) L3 cache. However, conventional SRAM or eDRAM are facing constraints of cell area and leakage energy consumption with technology scaling.

Recently, new non-volatile memory technologies, such as Spin-Torque Transfer RAM (STT-RAM), are being explored as potential alternatives of SRAM and eDRAM. It is attractive to use STT-RAM as L3 cache‡ because its small bitcell size for area savings and non-volatility for energy savings. Toshiba already revealed their plan in using STT-RAM to replace a 512KB SRAM L2 cache for low power purpose [1]. Another example is that replacing DRAM with STT-RAM in data centers can reduce power by up to 75% [2].

Unfortunately, while incorporating STT-RAM L3 cache has many advantages, one of the biggest issues is the asymmetric read/write of STT-RAM, which means that STT-RAM cache has much longer write latency and higher write energy [3], [4]. Considering STT-RAM write latency can be twice of SRAM’s, our experiment shows that directly replacing an SRAM L3 cache with an STT-RAM one can cause performance degradation up to 26%. Even worse, in a multi-core system, one process might frequently obstruct the STT-RAM L3 cache port and harm the performance of other processes running simultaneously.

Based on the above observations, we propose a sophisticated cache management policy for STT-RAM L3 cache. The contributions of this paper include:

- We define a type of workload characteristics called LLC-obstruction. Workloads with this characteristic have significant performance degradation in STT-RAM L3 based system, and they also negatively affect the performance of other processes.
- We design an obstruction-aware monitoring mechanism, OAM, which detects the behavior of LLC-obstruction processes during runtime.
- We propose an obstruction-aware cache management, OAP, which significantly improves performance and reduces energy consumption across different workloads with negligible hardware overhead.

Our experiment results show that the performance of a projected 4-core system with an 8MB STT-RAM L3 cache is improved by 14% on average and up to 42% after adopting OAP. This performance improvement offsets the performance loss caused by STT-RAM long write latency in directly replacing the SRAM L3 cache. In addition, OAP further reduces the STT-RAM L3 cache energy consumption by 64%. Therefore, OAP makes STT-RAM L3 cache a more attractive option for future microprocessor designs.

II. BACKGROUND AND MOTIVATION

A. STT-RAM Technology

STT-RAM is an enhanced variant of MRAM (Magnetic RAM). The basic storage element in STT-RAM is a magnetic tunnel junction (MTJ) as shown in Figure 1. Each MTJ is composed of two ferromagnetic layers. One has a fixed magnetization direction; the other has a free one and can change its direction. The relative direction of these two layers is used to represent a digital “0” or “1”. In STT-RAM technology, a switching current flowing from bitline to source line turns the cell into parallel state “0”; a switching current flowing from sourceline to bitline turns the cell into anti-parallel state “1”.

B. Using STT-RAM Last-Level Caches

Compared to conventional SRAM cache, the advantages of STT-RAM cache are smaller area and lower leakage power.
The cell size of STT-RAM is currently in the range of $13F^2$ [5] to $100F^2$ [6] where $F$ is the feature size. This cell size is much smaller than the SRAM size (e.g. $146F^2$ [7]). The small cell size of STT-RAM can greatly reduce the silicon area occupied by caches. In addition, STT-RAM has zero volatility. Previous work [8] shows the leakage energy can be as high as 80% of total energy consumption for an L2 cache in 130nm process. Thus, using STT-RAM last-level caches to eliminate standby leakage power is an attractive choice.

Table I shows the characteristics of a 4-bank 8-way 8MB SRAM cache and its STT-RAM counterpart. The estimation is given by NVSim [9], a performance, energy, and area model based on CACTI [7]. It shows that STT-RAM cache has much smaller area and leakage power than the one of SRAM. However, as also shown in Table I, the write latency of STT-RAM is around 2X longer than SRAM. This long write latency becomes a sensitive parameter in STT-RAM L3 cache and it causes the L3 cache port obstruction problem.

### C. Motivation

As the last-level cache, L3 is usually implemented using single-port memory bitcell due to the infeasible cost associated with multi-port bitcell designs. For example, building dual-port STT-RAM cell requires at least 4 transistor [10] (a 4X larger cell layout area), which is not affordable in large-capacity L3 designs. For single-port STT-RAM caches, an ongoing write operation can cause a long L3 cache port obstruction and delay all the following read operations that are on the critical path from the performance aspect. In addition, when an STT-RAM shard L3 is used in a multi-core system, an ongoing write from one core could block all the reads to the same cache bank from other cores, and it greatly reduces the overall system performance.

To illustrate this problem, we design two workload mixtures, mix1 and mix2 (see Section IV for details on simulation methodology). In mix1, we run 4 LLC-friendly workloads (omnetpp, hmmer, bwaves, and astar) on the 4-core system; In mix2, we only substitute workload omnetpp with cactusADM. Figure 2 shows the normalized IPC (instructions per cycle) of each process in these two mixtures using SRAM L3 and STT-RAM L3, respectively. As shown, for the mix1 group, the IPC of each process is reduced less than 5% when replace SRAM L3 with STT-RAM L3. However, for the mix2 group, not only the core running cactusADM has an IPC loss of 21% after using STT-RAM L3, but the IPCs of the other 3 cores are degraded by 8%-13% as well.

Thus, we define a special type of process characteristic called LLC-obstruction. Like cactusADM in Figure 2, if an LLC-obstruction process running in a multi-core system with STT-RAM L3 cache, it will not only experience a performance loss itself, but also negatively affects the performance of other processes running simultaneously on the system. Therefore, we need a new STT-RAM L3 cache management policy, and its task is to first identify any potential LLC-obstruction processes on-the-fly and then avoid the performance degradation caused by such LLC-obstruction processes.

### III. Obstruction-Aware Cache Management Policy

#### A. Heuristic Metric

The rationale behind adding another level of cache (e.g. L3) into the memory hierarchy is to provide fast-access memory resources so that workloads with good spacial and temporal locality can leverage these memory resources in L3 and avoid the time-consuming access to the off-chip DRAM main memory. However, not all the processes benefit from this feature.

Assuming that the behavior and characteristic of a process are not changed during a short period of running time, if we write data of this process into LLC, the expected execution time for one of following accesses can be estimated as Equation 1, in which, $P_{Rd}$ is the read/write ratio, $P_{Miss}$ is the miss rate of LLC. $T_{Rd}$, $T_{W}$, $T_{Mem}$ are the latency of LLC read, LLC write operation, and the average latency of main memory, respectively.

\[
T = P_{Rd} \times T_{Rd} + (1 - P_{Rd}) \times T_{W} + P_{Miss} \times (T_{Mem} + T_{W})
\]

(1)

We assume the cache uses fetch-on-write policy, which is a widely-used policy in modern cache designs.

![Fig. 1. The conceptual view of an STT-RAM cell.](image1)

![Fig. 2. Normalized IPC of systems with SRAM L3 and STT-RAM L3 for two groups of 4 mixed workloads, in which only the first core runs different workload and the others are the same.](image2)
For each access miss, new data need to be fetched from main memory and be written to LLC at first. Thus, the additional latency is added to the total delay $T$.

On the other hand, if the data of this process are not written into LLC, then following operations need to access main memory directly. Thus, the expected execution time for one access can be estimated as:

$$T' = T_{Mem}$$  \hspace{1cm} (2)

If writing data from one process to LLC cannot get any benefit in terms of system performance, which means $T > T'$, then we can get:

$$P_{Miss} > \frac{T_{Mem} - P_{Rd} \times T_{Rd} - (1 - P_{Rd}) \times T_{Wt}}{T_{Mem} + T_{Wt}}$$  \hspace{1cm} (3)

When the process characteristic satisfies Equation 3 during a period of runtime, it might cause intensive write operations to LLC so that writing its data into LLC will extend the execution time and degrade the system performance. We define this type of processes as LLC-obstruction processes.

We can also observe from Equation 3 that the possibility of LLC-obstruction is higher when STT-RAM LLC is used, because the miss rate threshold becomes smaller when $T_{Wt}$ is larger. It is also the reason that the performance degradation problem is more serious when we use STT-RAM LLC instead of its SRAM counterpart.

### B. LLC Obstruction Monitor

To detect LLC-obstruction processes during runtime, the obstruction-aware monitor (OAM) is devised and inserted before LLC (i.e. between the L2 and L3 caches). The OAM circuit is separated from the cache hierarchy, which allows OAM to independently obtain the L3 cache access information from each core.

A tunable parameter period is set in OAM, and it presents the period length of launching an LLC-obstruction detection. Every period is divided to two parts: sampPeriod, in which the cache works under the normal policy and OAMs collect statistics; exePeriod, in which L3 is managed by OAP using the detection result collected during the last sample period.

In sampPeriod, all processes are labeled as Non-LLC-obstruction. A per-core OAM collects statistic including: execution time currentTime, the number of read accesses RD, the number of write accesses WR, and the number of cache misses Miss.

At the end of sampPeriod, OAM calculates two parameters: the actual miss rate MissR and the obstruction threshold $OAP_{th}$.

$\text{MissR} = \frac{\text{Miss}}{(\text{RD} + \text{WR})}$  \hspace{1cm} (4)

$$OAP_{th} = \frac{T_{Mem} - (RD \cdot T_{Rd} + WR \cdot T_{Wt})/(RD + WR)}{T_{Mem} + T_{Wt}}$$  \hspace{1cm} (5)

According to Equation 3, if the result of MissR is larger than $OAP_{th}$, OAM labels this process as LLC-obstruction. Otherwise, OAM labels it as Non-LLC-obstruction. Then during the following exePeriod, this process is treated as the OAM detection result until the next period starts. The monitor algorithm is described in Algorithm 1.

### C. Architecture of OAP

Figure 3 shows a 4-core system with a 3-level cache hierarchy enhanced by OAP. Similar to previous works [11], [12], we assume processes are bound to cores. Each core has its own OAM to track L3 cache accesses requested by its private L2 cache and find out the LLC-obstruction processes. After the potential LLC-obstruction processes are detected, an OAP-controller in the shared L3 cache is responsible to prevent LLC-obstruction processes from accessing L3 cache and also make sure data coherence. The control flow of the OAP-controller is shown as Figure 4, and its functionality is described as follows:

- **L3 read hits**: L3 cache returns the data to the corresponding L2 cache.
- **L3 read misses**: L3 cache asks the data from main memory at first. When main memory returns the data,
the OAP-controller checks whether the read requester is LLC-obstruction. If it is, the returning data bypasses L3 and is directly forwarded to L2. Otherwise, the data is written into L3 as normal.

- **L3 write hits**: the OAP-controller checks if the write requester is LLC-obstruction at first. If it is, L3 invalidates the hit data block, and the write request bypasses L3 and is directly forwarded to main memory. Otherwise, the data is written into L3 as normal.

- **L3 write misses**: the OAP-controller checks if the requester is LLC-obstruction firstly. If it is, the write request bypasses L3 and is forwarded to main memory directly. Otherwise, the cache line is fetched and allocated at first and then the new data is written into L3 cache.

### D. Hardware Overhead

We evaluate the hardware overhead of the proposed OAP architecture using Synopsys Design Compiler with a 32nm CMOS library. According to the synthesis result, the OAM circuitry only incurs $0.05 \text{mm}^2$ area overhead, which is negligible compared to the L3 cache area (usually half of the total chip area). According to the energy analysis, the extra energy consumption per access is about 1.7pJ, which is also included in our simulations. In addition, it is straightforward to integrate the OAP judgment flow into the conventional cache controller using some bypass circuits.

For latency overhead, considering OAMs are separated from the cache hierarchy and OAP-controller only brings some branch decisions in the control flow, we do not add any latency overhead in our simulation settings.

### IV. EXPERIMENTAL RESULTS

In this section, we describe our experiment methodology, and evaluate the performance improvement and the energy reduction after applying OAP.

#### A. Experiment Methodology

We model a 1.5GHz 4-core out-of-order ARMv7 microprocessor using our modified version of gem5 [13]. Our modification to gem5 includes an asymmetric cache read/write latency model, a banked cache model, and a sophisticated cache write buffer scheme. Write-buffer technique is commonly used in conventional cache architecture to hide the performance penalty due to write operations, but the write buffer size cannot be too large because of its fully-associative look-up overhead [4]. Thus, the write buffer technique alone is not sufficient to deal with the performance degradation due to STT-RAM long write latency. We use an 8-entry write buffer in this work.

Simulation setting details are listed in Table II. All the circuit-level cache module parameters (e.g. read latency and write latency) are obtained from NVSim [9] and are consistent with Table I. The simulation workloads are from SPEC CPU2006 benchmark suite [14]. We simulate each experiment for at least 2 billion instructions. The OAM sampling period is set to be 0.1 million cycles and its launching period is set to be 10 million cycles.

We use IPC (instructions per cycle) as the performance metric and use geometric mean to average the performance of cores and derive the speedup metric ($\text{speedup} = \text{geomean}\left(\frac{IPC_i}{IPC_{\text{baseline}}}ight)$) which accounts for both fairness and performance [12].

#### B. Performance Speedup

Figure 5 illustrates the speedup over conventional STT-RAM L3 based system after adopting the OAP architecture. The first 8 groups run 4 duplicated processes, and the other 8 groups are workload mixtures as listed in Table III. It shows
that after adopting OAP on the STT-RAM L3 cache, the system performance is improved by 14% on average (up to 42%) compared to the conventional cache management policy.

Generally, the groups which have more LLC-obstruction processes in more periods benefit more from OAP architecture. Because OAM can quickly detect such processes during sampling periods, and OAP-controller skips their unnecessary writes to the STT-RAM L3 cache. For groups with mixed workload, the performance improvement comes from two respects. First, the performance of LLC-obstruction processes is improved because OAP skips the unnecessary writes in these processes and mitigates the penalty coming from the longer write latency. Second, the performance of concurrent processes is also improved since the obstruction on the L3 port is reduced and their requests to L3 can be satisfied more quickly. In addition, the available cache lines of L3 is increased for these well-behaved processes because the number of write operations from LLC-obstruction processes is reduced, thus it increases their hit rate and performance.

C. Compare to SRAM LLC

Adopting OAP architecture in L3 cache makes STT-RAM more competitive compared to SRAM. To evaluate its benefits, we simulate another baseline system with a shared SRAM L3 cache, which is configured as Table II.

Figure 6 shows the normalized IPC of the systems with normal SRAM L3, STT-RAM L3 and OAP STT-RAM L3. The result shows that compared to SRAM L3 based system, the conventional STT-RAM L3 degrades the system performance by 13.2% on average due to its longer write latency. However, after adopting the OAP architecture, the performance of STT-RAM L3 based system is improved significantly. Compared to SRAM L3, the performance degradation in OAP STT-RAM based system is only 0.7% on average.

Besides the performance improvement achieved by OAP, using STT-RAM L3 itself can bring energy savings. It is because the leakage power is dominant in L3 cache and the leakage power of STT-RAM is only about 1% of SRAM’s. In addition, adopting OAP can reduce the total energy consumption further, because it reduces the number of writes, which is a major source of the dynamic energy in STT-RAM caches. Figure 7 shows the normalized energy consumption of 3 different systems and each value is broken down to leakage energy and dynamic energy. Due to the leakage energy reduction, an 8Mb STT-RAM L3 can save around 90% total energy compared to an SRAM L3 cache with the same capacity. And after adopting OAP architecture, the energy of STT-RAM L3 is further reduced by 64% on average.

The final advantage of STT-RAM L3 cache is the silicon area reduction. It is not related to OAP but also important. Compared to an 8MB SRAM cache, an 8MB STTRAM cache can save 72.5% area as shown in Table I.

In summary, STT-RAM L3 enhanced by OAP makes itself a more attractive option in replacing SRAM L3 cache.

V. RELATED WORK

Some previous works focused on how to mitigate the performance penalty incurred by asymmetrically long write latencies of STT-RAM. Xu, et al. [15] proposed a dual write speed scheme to improve the average access time of STT-RAM cache. Smullen, et al. [16] and Sun, et al. [17] traded off the STT-RAM non-volatility to improve the write speed and the write energy. Sun, et al. [4] proposed a hybrid cache architecture with read-preemptive write buffers. An early write termination scheme was proposed to reduce the unnecessary writes to STT-RAM cells [18]. However, these works are all focused on modifying STT-RAM cell or cache architecture designs but not the cache management policy. There are also some previous works focused on cache management policies to improve the overall performance. Some tried to find the optimal partition and maximize system performance [11], [19], [20]. Xie, et al. proposed a pseudo-partitioning which combines targeted insertion and incremental promotion without enforcing the target partition [12]. Jaleel, et al. proposed a technique to identify the access pattern and prevent blocks with a distant reference interval from polluting the cache [21]. However, these works are focused on SRAM caches, where there is no read/write asymmetric problem, and they always allocate more than one cache way for each process. In this work, we tackled these two problems together and focused on how to improve the basic cache management policy for STT-RAM last-level caches.

VI. CONCLUSION

While the scaling of SRAM and eDRAM is constrained by cell density and leakage energy, STT-RAM as a new non-volatile memory technology is being explored as one of the potential alternatives of last-level caches. Despite STT-RAM
has higher density and smaller leakage energy, STT-RAM has much longer write latency compared to SRAM. The slower write operation could block the port of LLCs (e.g. L3 caches) for a longer time and degrade the system performance. Even worse, some write-intensive processes might interfere with other processes in a multi-core system. Thus, it is important to design an enhanced cache management policy that is optimized for STT-RAM LLCs.

In this work, we defined a type of characteristics called LLC-obstruction and proposed an obstruction-aware monitoring (OAM) mechanism that exploits the behavior of LLC-obstruction processes during runtime. In addition, we proposed OAP, an obstruction-aware cache management policy. By adopting OAP, a projected 4-core system with an 8MB STT-RAM L3 cache achieves an average 14% performance improvement (up to 42%) and further reduces the energy consumption by 64%. With the help of OAP, the STT-RAM L3 cache becomes a more attractive option for future microprocessor designs.

REFERENCES


