Shared LLC Pollution by Helper Threaded Data Prefetching on CMPs[*]

Min Cai, Zhimin Gu

Beijing Institute of Technology

Abstract

Helper threaded data prefetching on chip multiprocessors (CMPs), in its most basic form, utilizes the processing power of underutilized neighboring cores that communicate via a shared last level cache (LLC) to improve the performance of a single-threaded program running in a CMP. However, the multiple memory reference/miss streams that come from the main thread (MT) and the helper thread (HT) can inevitably stress and pollute LLC if no effective LLC content management techniques are employed.

In this paper, we present the methodology and mechanisms used for evaluating the cache pollution caused by helper-threaded data prefetching on CMPs to shed insights on improving the effectiveness and timeliness of the scheme. Firstly, the experimental framework for simulating helper-threaded data prefetching on CMPs is described. Secondly, the methodology to classify and quantify the contribution of HT requests to the MT performance is discussed. Results of memory-intensive benchmarks from Olden and CPU2006 show that: (1) there is notable cache pollution caused by helper-threaded data prefetching on CMPs; (2) there is room for improving the effectiveness and timeliness of helper-threaded data prefetching on CMPs.

Keywords: chip multiprocessors, helper threaded data prefetching, cache replacement, cache pollution

1 Introduction

TODO: background? motivation? our work: what? how? result?

TODO: Brief description about Helper Threaded Data Prefetching on Shared Cache Based CMPs.

MT = Main Thread (c0t0 here), HT = Helper Thread (c1t0 here). We assume here a two level cache hierarchy where L1 caches are private and the L2 cache is shared among all processor cores on a single chip.

TODO: Description about paper structure.

2 Related Work

TODO: previous work on prefetch taxonomies. Traditional coverage and accuracy metrics for h/w prefetches.

-> good, bad and ugly breakdown of h/w prefetches.

-> more fine-grained breakdowns of h/w prefetches.

any previous work on cache request breakdowns for helper threaded data prefetching?

TODO: a brief survey of recently proposed cache replacement policies. (1) victim selection; (2) insertion position or bypassing; (3) promotion on reference.

TODO: previous work on MLP based hardware mechanisms to improve the effectiveness of data prefetching. dynamic speculative precomputation (with chaining).

[*]Project (No. 61070029) Supported by the National Natural Science Foundation of China.

3 Quantifying the Contribution of HT Requests to MT Performance

Definition 1. Based on their contribution to the HT performance, for the HT requests to L2 cache shared by MT, HT and some other threads, the following taxonomy can be obtained:

# Good HT requests = # HT requests that hit by MT before evicted;
# Bad HT requests = # HT requests whose requested data are used later (or not used at all) by MT than the evicted data;
# Ugly HT requests = # HT requests - # good HT requests - # bad HT requests.

3.1 Tracking HT Request Victims: HT Request Evict Table

An HT Request Evict Table (ET) is attached to the L2 cache to track L2 lines that are evicted by HT requests and MT references to L2 lines brought or evicted by HT requests. ET has the same geometry as the L2 cache and uses the LRU replacement policy to track victim lines for each HT request per set. Specifically,

1. When a new HT request arrives at L2, an ET entry is allocated for tracking the victim line; and
2. when an HT requested line is evicted in L2, the corresponding ET entry for tracking the victim line is deallocated.

3.2 Identifying Good, Bad and Ugly HT Requests

There are two events to be considered for tracking HT requests' replacement victims and hits by MT: (1) when the L2 cache is filling a line; and (2) when the L2 cache is servicing a request.

3.2.1 Actions Taken on Filling an L2 Line

When the L2 cache is filling a line,

1. we should set HTRequest bit for the requested line if the requester is HT, and clear HTRequest bit for the victim line if the victim broughter is HT;
2. we should insert an entry in ET to track the victim line if the requester is HT and the victim line has invalid state or its broughter is non-HT;
3. we should remove the victim's corresponding ET line if the victim broughter is HT and the requester is non-HT;
4. for case 5:?.

The detailed actions are listed in Table 1.

No	Requester	Victim	Other Condition	Extra Cache Action	ET Action
1	HT	INVALID	None	set HTRequest for victim	insert NULL entry
2	HT	Non-HT	None	set HTRequest for victim	insert DATA entry for victim
3	HT	HT	None	update broughter HTAccess for victim	Update evicter HTAccess for victim HT request's Entry
4	Non-HT	HT	None	clear HTRequest for victim	remove victim's ET entry
5	Non-HT	Non-HT	∃htRequest in cache[set]	None	update victim for htRequest's Entry

Tab. 1: Actions Taken When Filling an L2 Line

3.2.2 Actions Taken on Servicing an L2 Request

When the L2 cache is servicing a request issued by MT,

1. under HT hit: we should clear the HTRequest bit for the requested line, and remove LRU in ET;
2. under ET hit: we should set LRU for the requested line in ET.

The detailed actions are listed in Table 2.

No	MT Hit	HT Hit	ET Hit	Cache Action	ET Action	Comments
1	No	No	No	Issue Miss	None	No changes
2	No	No	Yes	Issue Miss, clear HTRequest for ET entry	set LRU and remove entry	Bad HT request
3	No	Yes	No	Hit, clear HTRequest	remove HT request's entry	Good HT request
4	No	Yes	Yes	Hit, clear HTRequest, clear HTRequest for ET entry	remove HT request's entry, set LRU and remove entry	MT requested data evicted and requested back in by HT
5	Yes	No	No	Hit	None	No changes
6	Yes	No	Yes	Hit, clear HTRequest for ET entry	set LRU and remove entry	MT requested data evicted, requested back in by HT and hit to by MT
7	Yes	Yes	No	N/A	N/A	Impossible: cannot have identical tags in cache
8	Yes	Yes	Yes	N/A	N/A	Impossible: cannot have identical tags in cache

Tab. 2: Actions Taken When Servicing an L2 Request

4 MT Contribution Aware Reuse Distance Prediction Based Shared LLC Replacement with Bypassing

Use inter-MT/HT reuse distance prediction and HT prefetch quality information obtained above to make cache replacement decisions in order to keep useful prefetches on the shared L2, and for non-inclusive cache hierarchy, bypass useless and harmful prefetches directly to the L1 data cache private to the HT core.

5 MLP Based Chained Speculative Pre-Computation Mini-Threads for Tackling Late MT Prefetches

By utilizing the inherent memory level parallelism existing in chip multiprocessors, chained speculative pre-computation mini-threads can be constructed and spawned to prefetch in late yet potentially useful HT prefetches, thus improving the effectiveness of the HT prefetching scheme.

6 Experimentation

We use the yet-to-publish FleximJ simulator to evaluate different cache replacement strategies and dynamic pre-computation thread scheme proposed in previous sections. FleximJ is an execution-driven architectural simulator implemented in Java and running on Linux that supports application-only cycle-accurate modeling of multicore multithreaded architectures. It has preliminary support of simulating Pthreads based parallel workloads.

6.1 Simulation Setup

xx

6.2 Results

xx

7 Conclusion

TODO: our work: what? how? result? Further work?

Acknowledgment

This work was supported by the National Natural Science Foundation of China under the contract No. 61070029.