CS5222 Advanced Architectures

Project 3 Report

GHB G/AC Prefetcher

Nguyen Dang Phuc Nhat (A0184583U)

Table of Contents

1 Implementation Overview - GHB G/AC

For simplicity, I chose to implement the GHB G/AC prefetcher.

GHB G/AC architecture:

- 2 main components: GlobalHistoryBuffer and IndexTable
- GlobalHistoryBuffer: contains a FIFO queue of GHBEntry
    - GHBEntry: contains an address value _addr, a valid bit _valid, pointer to the next GHBEntry with the same address, and pointer to the previous entry in the FIFO
- IndexTable: contains a vector of IndexTableEntry
    - IndexTableEntry: contains an address value _addr, pointer to the latest GHBEntry of the same address _entry, and a variable _last_used
    - Replacement policy: I implemented the least-recently-used (LRU) for the IndexTable

Apart from the replacement policy of the IndexTable (which was not explicitly described by the authors), the rest of the implementation is in line with the description of the prefetcher.

From the Primer, the prefetcher operates when a cache miss occurs. The miss address will be added as a new GHBEntry to the head of the GlobalHistoryBuffer and looked up in the IndexTable. If there is a hit, the corresponding IndexTableEntry will update its _entry pointer to the newly added GHBEntry, and this GHBEntry will update is pointer to same address to whatever the IndexTableEntry is holding (which may even be NULL or an invalid address that was pushed out of the GlobalHistoryTable). If things go nicely, entries in the GlobalHistoryBuffer are accessed by a sequence of pointers, and their most previous entries in the FIFO contain the prefetch addresses.

The drawback of this prefetcher is glaring. It needs the program to reuse addresses (many times) in order to be useful. Some benchmarks are computationally heavy programs, which do not reuse addresses very much. It also fails to generalize patterns, compared to the delta-correlation approach. Furthermore, even with address correlation, the vanilla version GHB G/AC also cannot capitalize on miss streams' depths. The only tool to help with its coverage is prefetch degree, which may not be very useful if the addresses are vacated from the global buffer. The fact that the prefetch trigger is a cache miss could also translate to low coverage. Nesbit et. al's paper on GHB prefetchers also mentioned that GHB G/AC requires huge memory in order to be useful. It was not mentioned explicitly how huge the memory requirement is, but I guess it would be out-of-this-world. The paper also left out the performance of this prefetcher. Based on this, I predicted that it would be very low.

2 Benchmark Results

As per prediction, I found that GHB G/AC has subpar performance. I ran the benchmarks on a few variants of this prefetcher to see if anything can alleviate its shortcomings. Specifically, I ran the prefetcher with 512/4096/8192 table size (the same size for IndexTable and GlobalHistoryBuffer). Also, I changed a few lines of code to allow prefetching at any cache events. For cache hit, the prefetcher would prefetch based on the hit address. This may not be the best idea, but I think it's interesting to do an experiment.

IPC of prefetch-on-all-events versions:

Trace	**512-entry tables**	**4096-entry tables**	**8192-entry tables**
gcc	0.291329	0.291360	0.291373
GemsFDTD	3.445845	3.446584	3.446584
lbm	1.057604	1.057613	1.057613
leslie3d	0.981468	0.984902	0.984902
libquantum	3.148157	3.162956	3.163382
mcf	0.344241	0.344569	0.344840
milc	0.980482	0.980482	0.980482
omnetpp	2.189956	2.189652	2.189991

IPC of prefetch-on-miss versions:

Trace	512-entry tables	4096-entry tables	8192-entry tables
gcc	0.291329	0.291330	0.291231
GemsFDTD	3.444995	3.444995	3.444995
lbm	1.057604	1.057617	1.057616
leslie3d	0.985227	0.985514	0.985514
libquantum	3.148157	3.148157	3.148157
mcf	0.344332	0.344575	0.344573
milc	0.980482	0.980482	0.980482
omnetpp	2.191851	2.191851	2.192519

3 Discussion

From the IPC results, it seems that the increase in size of the tables does not matter at all. A big reason is that a low percentage of the prefetch requests is actually successful. In the simulator, the address prefetched must be from the same memory page as the trigger address in order for the request to succeed. I'm not sure why this is the case, but it surely limits the usefulness of GHB G/AC. Furthermore, the IPCs here are all roughly the same as the stream prefetcher that was pre-implemented. It occurs to me that GHB G/AC does not offer a competitive performance at all. A reason could be that addresses prefetched by GHB G/AC are not useful to the system at all, i.e. they are not used or replace useful cache lines.

I could see why Nesbit et. al. did not mention this prefetcher's performance. It's just pretty bad.