Bouquet of Instruction Pointers: Instruction Pointer Classifier based Hardware Prefetching

Samuel Pakalapati[1] and Biswabandan Panda[2]

[1]Intel Technology Pvt. Ltd. and BITS Pilani

[2]CSE Department, IIT Kanpur

ABSTRACT

Through this paper, we propose an instruction pointer classifier based hardware prefetching technique for the DPC-3. We use multiple instruction pointer based prefetchers that suit different access patterns and overall cover a wide spectrum of access patterns. Our classifier classifies instruction pointers at the L1 cache level and communicate the same to the L2 prefetcher. Our prefetching framework named Instruction Pointer Classifier based Prefetching (IPCP) provides 43.75% improvement for single-core and 22% for 25 selectively chosen multi-core mixes, respectively. IPCP demands a hardware overhead of 16.7KB per core.

1. INTRODUCTION

Hardware prefetching plays an important role in breaking the the fundamental "memory wall" problem. A hardware prefetcher observes the memory accesses (sitting beside a particular cache level) and predicts the future accesses based on a specific patterns that are driven by signatures like: instruction pointers (IPs), operating system (OS) page or a memory region. There are prefetchers that do not use any form of signatures, like next-line and offset based prefetchers.

Instruction pointer(IP) based prefetchers such as the IP-stride prefetcher [1] offers huge performance benefits that help in reducing the number of costly DRAM accesses. An IP-stride prefetcher, learns a recurring stride for an IP, and once it gets a minimum confidence on the learned stride, it starts prefetching with the learned stride. One of the advantage of using the IP based prefetchers is that it is less susceptible to IP reordering and random accesses across different memory regions, which may confuse a non IP prefetcher.

Our goal is to propose an IP based prefetcher that can cover the majority of access patterns. We find that each IP, can be classified into unique IP-classes, and the resulting classification could be used for better prefetching according to each class of IPs. For DPC3, we classify IPs into three classes: constant stride (CS), complex stride (CPLX) and stream (GS).

2. INSTRUCTION POINTER CLASSIFIER BASED PREFETCHING (IPCP)

In this section, we define and explain the rationale behind three different classes of IPs.

IP Constant Stride(CS): Consider the following access pattern (cache line aligned addresses) for an IP named IP_A: 0,2,4,6,8, with strides of 2,2,2,2. In this IP class, cache lines X, X+2, X+4 and so on are accessed by the same IP resulting in a constant stride pattern of 2. This is one of the more common patterns seen by IPs and can be prefetched using a simple IP-stride prefetcher. The prefetch coverage and prefetch accuracy for this pattern would be 100% if we classify this IP as IP for constant strides (IP-CS). We observe that there are accesses where a stride pattern skips a memory access in between. This could be because of a failed branch condition before the access or the accesses are filtered at the upper levels of cache. In that case, the constant stride access pattern would become X, X+2, X+4, X+8. Note that X+6 is missing. A simple 2-bit counter per IP entry (with values from zero to three) can help to maintain *hysteresis* as mentioned in the original IP-stride prefetcher [1] and we can keep on prefetching until this confidence falls below a certain threshold (less than two).

IP Complex Stride(CPLX): Consider this access pattern by IP_B: 0,3,6,10,13,16,20 with strides of 3,3,4,3,3,4.
Here we see a complex pattern by IP_B. A simple constant stride prefetcher would give 66% coverage assuming these strides are cold start strides, since it would be unable to predict stride 4. Also if the stride pattern is 1,2,1,2,1,2, a simple IP-stride prefetcher would lack the confidence to prefetch any stride since the two strides compete for the same entry in the stride table. Here, its coverage would be zero. These patterns are classified as complex strides. Taking cue from the SPP prefetcher proposed in DPC-2[2], we create a signature of deltas seen by an IP and use it to index into a delta prediction table (DPT). It may be argued that a constant stride pattern is a subset of a complex stride pattern, hence removing the need to classify an IP separately as CS. But we identify two benefits to do so. Firstly, a CS prefetcher is more resilient to random mismatches in the patterns due to the hysteresis counters. A CPLX prefetcher would just start constructing a new path every time such a random access occurs. During this time it would be unaware of what to prefetch next. Secondly a CPLX prefetcher has to hash the signature (a

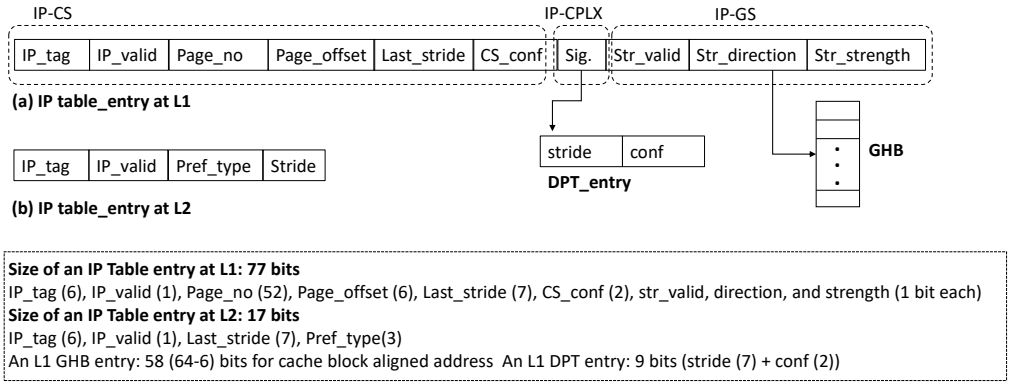

Figure 1: An entry in L1 and L2 IPCP tables. Pref-type has five different prefetch types (None, GS, CS, CPLX, and NL) classified at the L1. Page offset is the cache line offset (ranging from 0 to 63) within an OS page.

signature created based on the deltas seen so far) and look up the DPT for every consecutive access. This could be costly in terms of latency whereas a CS prefetcher would simply add the delta to the next access to be prefetched. Note that delta is nothing a but a stride between two cache line aligned addresses.

IP Global Stream (GS): A global stream is a set of consecutive cache line aligned accesses observed independent of an IP, for example an access stream of 0,1,2,3,4,5,6,7,8,9. While this stream can be prefetched using using the CS or the CPLX class, looking at the global pattern we see that there is one IP that triggers a global stream and many IPs follow the global stream (makes the access stream independent of the IP). So, it makes more sense to prefetch the global stream as opposed to the IP-correlated stream since it preserves the global order of accesses and results in much better timeliness. So if an IP belongs to multiple classes, we prioritize the global stream class over others.

Temporal and Irregular IP (TIP): There are irregular accesses, which do not follow any pattern, simple or complex, but recur. ISB prefetcher [3] suits these access patterns, where IP-correlated streams are stored and prefetched when required. We do not explore this prefetching in this paper due to the lack of access to off-chip memory and the translation look-aside buffer (TLB), as per the DPC3 rules, without which the memory cost of storing these accesses on chip becomes prohibitively high. Integrating the ISB as a temporal class IP would result in higher coverage as both irregular and regular accesses would be covered.

3. DESIGN OF IPCP

We implement IPCP at two cache levels: L1 and L2. We do not implement it at the LLC as we do not see any considerable benefit. The prefetch requests issued into L2 and L1 are also filled into last-level cache (LLC). On a demand access (along with prefetch hits) at the L1, IPCP stores the corresponding IP in an IP table indexed using the IP bits. The consecutive accesses by the same IP are used to calculate the strides and classify the IP into one of the three classes. The access stream at the L2 is now jumbled since it consists of prefetch requests and demand misses from the L1. Thus we cannot simply train on the L1 misses like we used to in the absence of an L1 prefetcher, since some of the misses are converted to hits due to L1 prefetching. This corruption of the stream makes pattern matching at the L2 under the present circumstances very difficult. Hence we use the L1 prefetch requests to communicate the IP classification information to the L2 prefetcher (through the metadata feature available in DPC-3). So, on a prefetch access to L2 (because of L1 prefetching), the L2 prefetcher issues prefetch requests according to the IP classification done at the L1. On a demand access at the L2, the L2 prefetcher issues prefetch requests based on the metadata information only.

3.1 IPCP at the L1

Figure 1 (a) shows the IP table entry at the L1 prefetcher that is used by IPCP. IPCP uses a table of 1024 entries at L1.

Fields of interest for CS class: The IP table entry consists of various fields for three different classes of IPs. The IP table is indexed and tagged by an IP. Each entry has a *last stride* entry corresponds to the constant stride (CS) class. A confidence counter *conf* is incremented every time the same stride is seen. It is used to determine whether to prefetch using the constant stride. The entry stores the last OS page (*last page*) seen by the IP and the *page offset*. The offset is used to calculate the stride between two IP accesses. The page information is used for page boundary learning and calculating the stride when a new page is seen.

Fields of interest for CPLX class: The *signature* is part of the CPLX class, points to the last delta(s) seen by the IP. The new delta is then hashed with the signature to lookup to a table called delta prediction table (DPT) to predict the next delta. DPT consists of 4096 entries and is indexed using the 12-bit signature stored in the IP table. Each entry stores the delta pointed to by the signature and a 2-bit saturating confidence counter. Every time the same delta is pointed to by a signature, the confidence counter is incremented by one and decremented otherwise. This delta is hashed to the signature and the DPT is looked up to issue prefetch requests.

Fields of interest for GS class: The fields related to the global stream class are *stream_valid*, *stream_strength* and *stream_direction* bits correspond to the global stream (GS) class. We define a global stream as a sequence of contiguous cache line aligned addresses. When the stream valid bit is set, the IP is known as the trigger IP of the global stream. The global stream IP class uses a small Global History Buffer(GHB)[4] that keeps track of the last 16 cache

line aligned addresses seen in the global stream (independent of any IP). We say an IP is a global stream IP, when more than half of the entries correspond to the same stream (which is measured in both directions from the current access). If this number is greater than 3/4th of the entries then the IP is strongly classified as a global stream. This means the IP is permanently classified as *GS* class and no further checking of the GHB is necessary to verify its class in the future. The direction bit is used to decide the next *n* cache lines in the +ve or the -ve direction. The `strength` bit indicates the number of accesses in the global stream covered by the stream, and refereed to as a strong stream if the stream covers 3/4th of the GHB entries.

Indexing into the IP table: The IP tracking table at the L1 is indexed by the last 10 bits (lsbs) of the IP. It stores the next six IP bits as the tag bits. As there can be collisions between IPs matching to the same table entry, we use a valid bit to maintain hysteresis. When an IP is encountered for the first time, it is recorded in the table and the valid bit is set. When another IP maps to the same entry, the valid bit is reset but the previous entry remains. If the valid bit is zero (reset) when a new IP is seen then the table entry is allocated to the new IP and the valid bit is set again. This ensures that at least one of the two competing IPs is tracked in the table. We do not use any replacement policy and instead focus on indexing, since it would induce too much latency into the critical path, especially at the L1 level.

The case of no-IP (next-line prefetching): In case a demand access does not fall into any of the three classes (CS, CPLX, and GS), we use the next-line (NL) prefetcher. However, the usage of NL prefetcher can be detrimental to performance especially in case of irregular access patterns. So, we also use a few other structures to make adaptive decisions. We calculate the L1 misses per kilo cycles (MPKC) per core. A miss counter counts the number of L1 misses. After every 256 (determined empirically) demand misses, the number of cycles elapsed is calculated and the MPKC is calculated. Since we cannot afford useless prefetches when the MPKC is too high, we turn off NL prefetching at the L1 and L2. A *speculative_NL* bit is set for each cache level, when the MPKC is below a certain threshold (15 for single core and five for multi-core) and reset otherwise. NL prefetching is ON only when this is set.

Page boundary learning: IPCP stores the page offset (cache line offset within an OS page, for a 4KB page, a offset can have values from 0 to 63) of the current access. It is used to calculate the stride between the two accesses by the same IP. The page of the current access is checked against the OS page stored in the IP entry. If the pages match, the stride is from the same page. If not, 64 (the number of cache lines within the page) is added to a negative stride or subtracted from a positive stride. This new stride is used to train the CS and CPLX classes, incrementing or decrementing the confidence of the respective classes.

3.2 IPCP at the L2

The IP table at the L2 is only used for book-keeping purposes. The metadata obtained from the L1 prefetcher is used to classify the IPs as belonging to one of the four classes: CS, CPLX, GS, and NL (in case an IP does not belong to CS, CPLX, and GS). IPCP at the L2 uses an IP table of 1024 entries and is indexed by the last 10 bits (lsbs) of the IP and each entry contains a 6-bit IP tag, an IP-valid bit, a 2-bit type field (based on the metadata information, there are four possibilities) and a 7-bit stride. Figure 1(b) shows the IP table entry at L2. On a demand access at the L2, the L2 prefetcher consults the L2 table that is populated based on the metadata information communicated by the L1 prefetcher.

There is also a *speculative_NL* bit at the L2 which is constantly updated using the metadata from the L1. The NL prefetcher at L2 issues prefetch requests when this bit is set.

Metadata Decoding at L2: When a prefetch is issued from L1, some form of metadata is sent along with it in DPC-3. This is used to communicate with the lower level prefetchers. In our case, the metadata contains the *stride* for each class, the type of IP class (pref-type in the L2 entry) and the MPKC information to decide whether to issue next line prefetches (in the form of speculative NL bit). The metadata does not contain the IP because the IP of the request is anyway passed to the L2 prefetcher. The L2 prefetcher decodes the stream of L2 accesses and the metadata sent by the L1 prefetcher, and stores it in the L2 IP table. When an IP is seen at the L2, prefetch requests are issued according to the stored metadata, regardless of whether it is a demand or prefetch request. IPCP at the L2 does not issue prefetch requests for the CPLX class. CPLX class at the L2 does not yield any benefits and even caused performance degradation in some of the benchmarks.

3.3 Working of IPCP and priority of IP Classes

Priority of classes: IPCP uses the following priority: GS, CS, CPLX, and then NL. At a particular instant, if an access belongs to multiple IP classes then this priority order is used. Note that, IPCP does not access the table multiple times to find out the class a particular demand access, because all the information is stored as part of a single entry. All the classes can be checked concurrently and finally the highest priority class is used, in case of a tie.

Overall, on a demand access, the *stream_valid* bit is checked first. If the IP belongs to the GS class, the next *six* addresses are issued according to the *stream_direction* bit. In case the IP is a not "strong-stream" IP, the GHB is checked to see if the IP still belongs to a global stream. Only the valid and direction bits are required to check for global stream prefetching. Each access is added to the GHB if it is not already found in it, i.e., each GHB entry is unique. If the IP does not belong to the GS class, the CS entries are checked, concurrently. If the confidence is high enough (greater than one in our case), the stride is added recursively to the current cache line address until the number of prefetch requests specified in the preferch degree (three in our case) are issued. If not, the pattern is complex and the DPT has to be queried to get the next delta.

The stride obtained previously is added to the signature according to the equation: *signature = (signature « 1) ^ stride* Note that we shift the signature by a single bit so that we can accommodate highly complex stride pattern. Thus a pattern can produce many signatures, but we do not observe too many collisions in the DPT because there are not many CPLX IPs at the same point of time. The signature points to the next delta, and if the confidence is high enough(> one in our case), the delta is added to the cache line to produce the

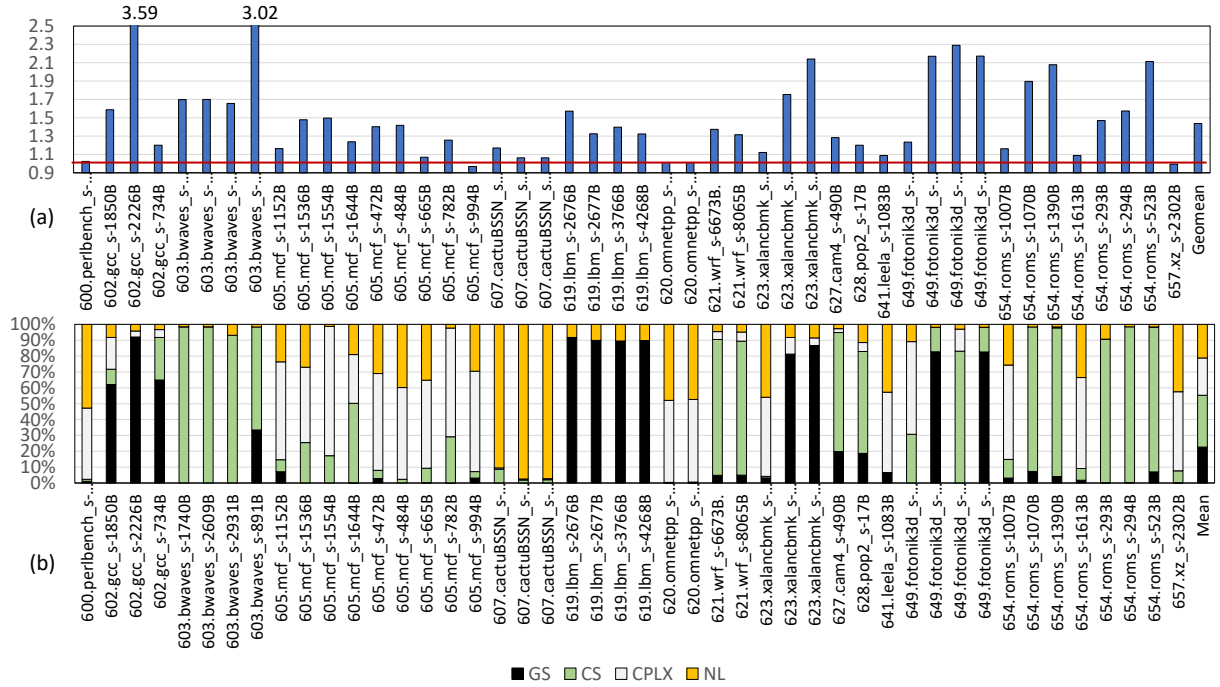

Figure 2: (a)Normalized single-core IPCs. (b) Distribution of prefetch requests based on the IP class.

	entry-size × entries	
IP table	77×1024 (L1) + 17×1024 (L2)	12.03KB
DPT table	9×4096	4.6KB
GHB table	16×58	928 bits
Prefetch degrees	6 bits (L1) and 8 bits (L2)	14 bits
Others	MPKC (8),cyle-counter(64), miss-counter(8), MPKC-threshold(4), speculative-NL bit at L1 and L2 (2)	86 bits
Total		16.7KB

Table 1: Hardware Overhead with IPCP at L1 and L2.

prefetch address. If the confidence is greater than zero, the delta is added to the signature using the above equation and the next prediction is made. This look-ahead continues until the prefetch degree (three in our case) is reached.

If critical path is an issue then CPLX class can be moved to the L2 prefetcher.Note that this is applicable only to the CPLX class.Prefetching with the other classes simply involves adding the delta to the current access to predict the next access. If all the previous classes are not satisfied, IPCP turns into a NL prefetcher depending on the *speculative_nl* bit.

Prefetcher Aggressiveness for each class: IPCP issues prefetch requests with a prefetch degree of three for CS and CPLX classes at L1 and six for the GS class. For L2, IPCP uses a prefetch degree of four for GS class. For the CS class, IPCP uses a degree four if the MSHR occupancy is less than one half of the MSHR entries, else it uses a degree three. With IPCP, the prefetch degree for multi-core is less than that of single core, since there is a lot of contention at the DRAM controller. We throttle the degrees of GS from six to four, three to two for CPLX and CS, at the L1. At the L2, we use the degree two for both CS and GS, respectively.

Hardware Budget: A self-contained Table 1 shows the hardware overhead, which is 16.7KB per core.

4. RESULTS

The speedup obtained by IPCP for single core is 43.75% with maximum improvements of more than 3X coming from `gcc` and `bwaves`. Out of all the traces only one trace `mcf-994` shows a non-negligible performance degradation of more than 2%. For a representative mix of multi-core workloads, we observe a speedup of 22%. Figure 2 (a) shows the normalized IPC improvements and Figure 2 (b) shows the distribution of prefetch requests based on their respective classes. On average, IPCP has used all the classes equally.

5. CONCLUSION

This paper proposed a bouquet of instruction pointers (IPCP) that is used to prefetch various kinds of access patterns at the L1 and L2 caches. IPCP provides an average improvement of 43.75% for single-threaded traces.

6. REFERENCES

[1] S. P. Vanderwiel and D. J. Lilja, "Data prefetch mechanisms," *ACM Comput. Surv.*, vol. 32, pp. 174–199, June 2000.

[2] K. et al., "Path confidence based lookahead prefetching," in *(MICRO 2016)*, pp. 1–12, 2016.

[3] A. Jain and C. Lin, "Linearizing irregular memory accesses for improved correlated prefetching," in *(MICRO 2013)*, pp. 247–259, 2013.

[4] S. Palacharla and R. E. Kessler, "Evaluating stream buffers as a secondary cache replacement," in *ISCA '94*, pp. 24–33.