Hardware Prefetcher Aggressiveness Controllers: Do We Need Them All the Time?

Anuj Mishra and Biswabandan Panda,
Department of Computer Science and Engineering, IIT Kanpur, India

1. ABSTRACT

Hardware prefetching is a well known latency hiding technique for improving performance. A hardware prefetcher predicts future memory references and brings data to cache before processor demands it. However, in case of many-core systems, prefetchers can increase shared resource contention such as DRAM bandwidth contention, degrading overall system performance. Research on controlling aggressiveness (that controls prefetch degree and distance) of prefetchers has focused on heterogeneous workloads having a mix of prefetch friendly (applications that get benefit with prefetching) and unfriendly (applications that do not get benefit with prefetching). But a large number of server workloads are multi-threaded and homogeneous in nature. We showcase, prefetch aggressiveness controllers that perform well in case of heterogeneous workloads by classifying and grouping applications, provide marginal utility in case of homogeneous workloads. Our findings show that there is no need of grouping or clustering mechanisms for controlling aggressiveness. Instead, we make a case for a simple online profiler that can decide whether to keep prefetchers ON or OFF.

2. MOTIVATION

The number of cores in current generation servers have increased substantially. However, the DRAM bandwidth has not been able to keep up with this increase [1]. For example, a recent Intel Xeon [1] of 28 cores (56 hyper-threading threads) has support for only six DRAM channels, only one DRAM channel per nine threads. This leads to an increase in contention for DRAM bandwidth among the cores. State-of-the-art prefetchers like IPCP[13], SPP[12] and Bingo [4] show significant improvement in single core performance but also increase the DRAM traffic because of additional prefetch requests. The additional traffic because of prefetching, increases the contention at all shared resources (on-chip network, last level cache (LLC), DRAM bandwidth). However, recent studies [5] show LLC contention is not a problem with large LLC. So, we focus primarily on the effect of DRAM bandwidth on the effectiveness of hardware prefetching. Figure 1 shows the performance improvement provided by different state-of-the-art prefetchers such as IPCP[13], SPP[12], and Bingo[4] with high (4 channels, one channel for four cores) and low(2 channels, one channel for eight cores) DRAM bandwidths for a 16 core system. We use a per channel bandwidth of DDR4-3200 (25.6 GB/s). As we can see, for prefetch friendly applications, all the prefetchers experience performance reduction in case of low bandwidth

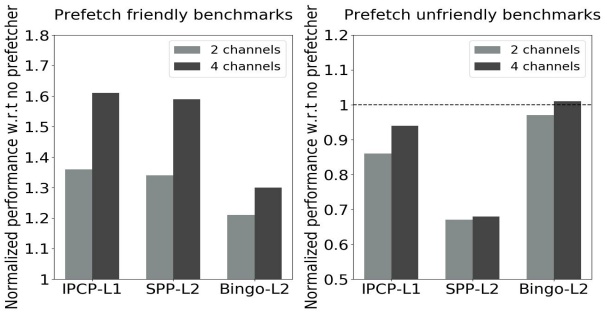

Figure 1: Performance improvement with different state-of-the-art prefetchers for a 16-core system having three-level cache hierarchy with high (4 channels) and low(2 channels) DRAM bandwidths. IPCP-L1: IPCP at the L1, SPP/Bingo-L2: SPP/Bingo at the L2.

(2 DRAM channels) and in case of prefetch unfriendly applications, there is a performance degradation irrespective of available DRAM bandwidth. Prefetch aggressiveness controllers tackle this problem by controlling prefetch degree and prefetch distance. However, prior works [7], [11], [14] [10] [16] [8] [15] [9] on prefetch aggressiveness control focus primarily on heterogeneous mixes (multicore system with each core running a different application), propose changes in hardware for classifying applications into clusters (groups) based on prefetch behavior. Although the mechanisms proposed in prior works improve performance for heterogeneous workloads, they have marginal utility for homogeneous workloads (all cores running the same application) that are common in server environments. Also, majority of the mechanisms are proposed for simple stride and stream based prefetchers but fails to deliver performance benefit over state-of-the-art prefetchers as they have in-built mechanisms for aggressiveness control based on prefetch accuracy and coverage.

In the next section, we present detailed analysis of our idea and conclude that all the prior aggressiveness controllers have minor utilities in terms of performance improvement and we can get similar or better performance from prefetching using a naive performance based profiler.

3. DO WE NEED CLUSTERING?

Prior prefetch aggressiveness controllers use grouping (clustering) of applications for controlling prefetch degree and prefetch distance. We showcase that there is no need of grouping (clustering) with homogeneous workloads. We extend ChampSim[3] (a trace-based simulator used in Data

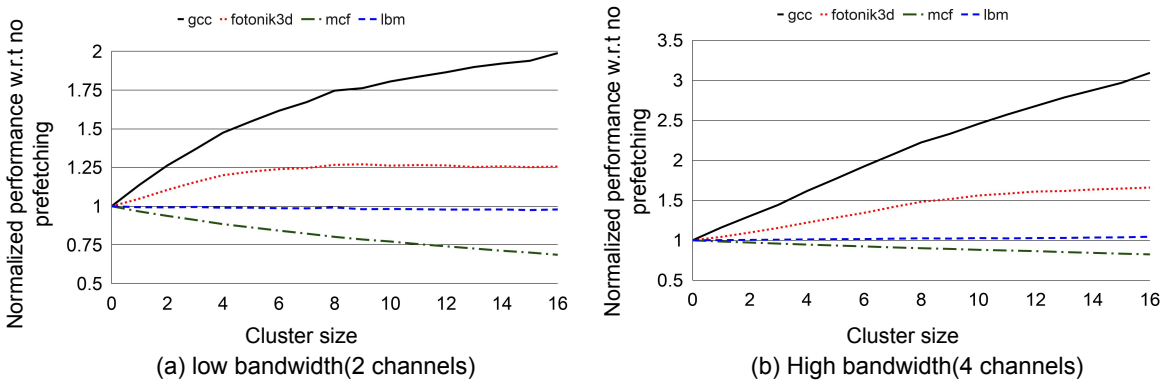

Figure 2: Performance improvement with hardware prefetching for different cluster sizes for a 16 core system. Higher the better.

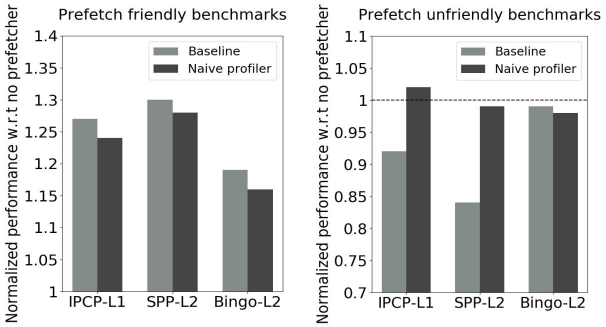

Figure 3: Performance improvement with a naive profiler for a 16 core system with low DRAM bandwidth.

Prefetching Championship[2]) for many-core system with detailed DRAM bandwidth, on-chip-interconnect, and sliced LLCs and compare the effectiveness of different clustering mechanisms. With large LLCs, and scalable high-bandwidth interconnect, off-chip shared DRAM bandwidth becomes the bottleneck on many-core system. We show the results for a 16 core simulated system in SPEC[6] rate mode. We select four SPEC 2017 benchmarks that are representative of different performance trends that we observe across all memory intensive SPEC 2017 benchmarks.

Size of the cluster (number of cores with hardware prefetching ON) is an important factor in prior prefetch aggressiveness controllers. Size of the cluster determines the (i) amount of time for which a core (application) gets benefit from prefetching as a small cluster size would mean more clusters whereas a large cluster would mean few clusters), and (ii) the DRAM bandwidth congestion caused by prefetch requests.

In Figure 2, we show the performance improvement on a 16-core system, with IPCP prefetcher [13] for all cluster sizes with low and high DRAM bandwidths. Cluster size of 16 means that all the 16 applications running on 16 cores, have their respective prefetchers ON whereas cluster size of zero means, all the prefetchers are OFF. We mainly concentrate on Figure 2 (a), where DRAM bandwidth is scarce and it is a good indicator of recent Intel servers [1]. We observe that on varying size of the cluster (number of applications with prefetching ON), different applications show different behaviors, for instance, prefetch friendly applications (such as `gcc`) shows performance improvement with an increase in cluster size across all bandwidths, while prefetch unfriendly applications (`mcf`) perform better with lower cluster sizes. We observe an interesting behavior for `fotonik`, unlike other applications, we see a fluctuation in performance improvement with higher cluster sizes, which is a result of an interplay between the two attributes (i) and (ii) that we discussed above.

We observe from Figure 2 that we get the best performance for prefetch friendly benchmarks when cluster size is equal to number of cores (16 in our case) (except for `fotonik` that performs slightly better with cluster size three in case of low bandwidth), which is equivalent to having all prefetchers on. While for prefetch unfriendly benchmarks, the performance peaks are are always at cluster size zero, which suggests having prefetcher OFF for all the cores provide best system performance.

This binary decision of having all prefetchers ON or OFF based on prefetch behavior can be done by using either offline or online profiling. We showcase that a naive profiler that is light-weight and online in nature is highly effective in improving system performance. Our profiler has a small training phase followed by an execution phase, where in the training phase, the profiler learns whether it is beneficial to turn the prefetcher ON or OFF. Figure 3 shows the performance improvement by prefetching with our naive profiler compared to baseline (having all prefetchers on) for a 16-core simulated system with low bandwidth(2 channels). We get the primary benefit for prefetch unfriendly applications. On average, we observe 10 and 15 % improvement for IPCP and SPP, while Bingo does not show any improvement because of its conservative in-built prefetch throttling mechanisms. For prefetch friendly benchmarks, we get performance similar to baseline.

4. CONCLUSION

We highlight the problem of existing prefetch aggressiveness controllers on many-core systems. Through experiments on a 16-core simulated system with different DRAM bandwidths across different cluster sizes, we conclude that there is no utility of clustering used in prefetch aggressiveness controllers for homogeneous workloads and we could get maximum benefit by using a naive performance based profiler that can either keep all the prefetchers ON or OFF.

5. REFERENCES

[1] https://ark.intel.com/content/www/us/en/ark/products/205684/intel-xeon-platinum-8380hl-processor-38-5m-cache-2-90-ghz.html.

[2] https://dpc3.compas.cs.stonybrook.edu/.

[3] https://github.com/ChampSim/ChampSim.

[4] M. Bakhshalipour, M. Shakerinava, P. Lotfi-Kamran, and H. Sarbazi-Azad. Bingo spatial data prefetcher. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 399–411, 2019.

[5] Rahul Bera, Anant V. Nori, Onur Mutlu, and Sreenivas Subramoney. Dspatch: Dual spatial pattern prefetcher. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52, page 531–544, New York, NY, USA, 2019. Association for Computing Machinery.

[6] James Bucek, Klaus-Dieter Lange, and Jóakim v. Kistowski. Spec cpu2017: Next-generation compute benchmark. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, ICPE '18, page 41–42, New York, NY, USA, 2018. Association for Computing Machinery.

[7] Eiman Ebrahimi, Chang Joo Lee, Onur Mutlu, and Yale N. Patt. Prefetch-aware shared resource management for multi-core systems. In *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, page 141–152, New York, NY, USA, 2011. Association for Computing Machinery.

[8] Eiman Ebrahimi, Onur Mutlu, Chang Joo Lee, and Yale N. Patt. Coordinated control of multiple prefetchers in multi-core systems. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, page 316–326, New York, NY, USA, 2009. Association for Computing Machinery.

[9] Wim Heirman, Kristof Du Bois, Yves Vandriessche, Stijn Eyerman, and Ibrahim Hur. Near-side prefetch throttling: Adaptive prefetching for high-performance many-core processors. In *Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques*, PACT '18, New York, NY, USA, 2018. Association for Computing Machinery.

[10] V. Jimenez, A. Buyuktosunoglu, P. Bose, F. P. O'Connell, F. Cazorla, and M. Valero. Increasing multicore system efficiency through intelligent bandwidth shifting. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 39–50, 2015.

[11] Victor Jiménez, Roberto Gioiosa, Francisco J. Cazorla, Alper Buyuktosunoglu, Pradip Bose, and Francis P. O'Connell. Making data prefetch smarter: Adaptive prefetching on power7. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, PACT '12, page 137–146, New York, NY, USA, 2012. Association for Computing Machinery.

[12] Jinchun Kim, Seth H. Pugsley, Paul V. Gratz, A. L. Narasimha Reddy, Chris Wilkerson, and Zeshan Chishti. Path confidence based lookahead prefetching. In *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-49. IEEE Press, 2016.

[13] Samuel Pakalapati and Biswabandan Panda. Bouquet of instruction pointers: Instruction pointer classifier-based spatial hardware prefetching. In *Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture*, ISCA '20, page 118–131. IEEE Press, 2020.

[14] B. Panda. Spac: A synergistic prefetcher aggressiveness controller for multi-core systems. *IEEE Transactions on Computers*, 65(12):3740–3753, 2016.

[15] Biswabandan Panda and Shankar Balachandran. Caffeine: A utility-driven prefetcher aggressiveness engine for multicores. *ACM Trans. Archit. Code Optim.*, 12(3), August 2015.

[16] Aswinkumar Sridharan, Biswabandan Panda, and Andre Seznec. Band-pass prefetching: An effective prefetch management mechanism using prefetch-fraction metric in multi-core systems. *ACM Trans. Archit. Code Optim.*, 14(2), June 2017.