Improving the prefetching performance through code region profiling

Martí Torrents[1], Raul Martínez[1], Carlos Molina[2]
[1]UPC-BarcelonaTech, [2]Universitat Rovira i Virgili
{martit,raulm}@ac.upc.edu, carlos.molina@urv.net

Abstract- *In this work, we propose a new technique to improve the performance of hardware data prefetching. This technique is based on detecting periods of time and regions of code where the prefetcher is not working properly, thus not providing any speedup or even producing slowdown. Once these periods of time and regions of code are detected, the prefetcher may be switched off and later on, switched on. To efficiently implement such mechanism, we identify three orthogonal issues that must be addressed: the granularity of the code region, when the prefetcher is switched on, and when the prefetcher is switched off.*

I. INTRODUCTION

An imbalance in technological improvements in recent years has led to an increasing gap between processor and memory speeds. One way to address this problem is to use latency hiding techniques such as prefetching. This mechanism tries to predict which data the processor will require in the near future and bring it to the nearest cache level before it is needed by the application. Prefetching is a key technique employed by almost all current commercial high-performance processors in at least one but typically all levels of their memory hierarchy [1].

However, the prefetcher works as a double-edged sword because, if the prefetcher does not work properly, and too many of the prefetching requests are useless [1], the mechanism will not increase performance but still may contribute to increment network latency, power consumption, and cache pollution [2]. This is particularly critical in CMP environments, were non-accurate prefetchers may perform important slowdowns since the network becomes another key resource, [2].

As a general rule, the prefetcher produces speedup as long as it generates a certain number of useful prefetch requests[3] respect the total number of requests generated by the prefetcher (i.e. the accuracy [4] is higher than a certain threshold that depends on the application and the prefetcher)..Moreover, in the scope of CMP environments, we have observed that, the implementation of the prefetcher in a distributed and shared memory system is not trivial. There are some challenges [3] that must be handled in order to not increase the cases where the prefetcher does not work accurately and produces slowdowns.

[1] A **useless prefetch** is a request injected in the cache by the prefetcher and evicted from it without being used by the core.

[2] Prefetch requests **pollute** the cache when they replace data that is still required by the program.

[3] A **useful prefetch** is a request injected in the cache by the prefetcher that is actually requested afterwards by a regular memory request

[4] The **accuracy** is measured as the number of useful prefetch divided by the total prefetch requests.

II. PROPOSAL

Although the prefetcher may contribute to global speedup, there may be periods of time and regions of code during the execution where the prefetcher may introduce slowdown. Our proposal aims to improve the global performance of the prefetcher by switching it off when it does not work properly, and switch it on again when it would provide actual performance improvements. Therefore, out target is to reduce the congestion of the system, the pollution that the prefetcher may produce in the cache module, and to carry out a better power management.

In order to achieve these objectives, there are several issues that must be properly addressed in order to obtain the maximum benefit from this technique.

1. **Code region granularity**. There are several types of code region to be considered: single instructions, basic blocs, superblocks, inner loops, outer loops, the whole dynamic code, etc. To implement this kind of mechanism, it would be mandatory to get prefetching statistics when executing code dynamically. Techniques such as [4] may be useful.
2. **Switching off the prefetcher**. Several statistics can be analyzed in order to switch off the prefetcher. In this study, we will focus on a particular prefetching statistic as the accuracy and also on cache miss ratios.
3. **Switching on the prefetcher**. The problem of switching on again the prefetcher is that, when the prefetcher is switched off, we cannot collect prefetching statistics. Therefore, we will focus on cache miss ratios and timeout mechanisms

ACKNOWLEDGMENT

This work has been partially supported by the Spanish Ministry of Science and Innovation (MCI) and FEDER funds of the EU under the contracts TIN2010-18368 and TIN2013-47245-C2-2-R, and the Generalitat of Catalunya under grants 2009SGR1250.

REFERENCES

[1] M. Torrents, et al. “Comparative study of prefetching mechanisms”. Jornadas del Paralelismo, Valencia (Spain), 2012.

[2] M. Torrents, R. Martínez, C. Molina. “Network Aware Performance Evaluation of Prefetching Techniques in CMPs”. Simulation Modeling Practice and Theory (SIMPAT), 2014.

[3] M. Torrents, et al. “Prefetching Challenges in Distributed Memories for CMPs”, In Proceedings of the International Conference on Computational Science (ICCS'15), Reykjavík, (Iceland), June 2015.

[4] R. Martinez, J. Codina, E. Gibert, P. Lopez, M. Torrents, et al. “Profiling asynchronous events resulting from the execution of software at code region granularity”. U.S. Patent Application 13/993,054, 2011.