A Survey of Different Prefetching Techniques on NoC

Manju B Menon
M. Tech Scholar
Department of Information Technology
Govt. Engineering College, Barton Hill
manjubmenon.manju@gmail.com

Josna V R
Assistant Professor
Department of Information Technology
Govt. Engineering College, Barton Hill
Josna.chandu@gmail.com

Abstract— Prefetching is an effective technique on NoC. This technique is used to avoid a well-known problem on multicore processors. Prefetching is an appropriate technique for decreasing memory latency. Data prefetching is fetching the data from memory and placing it into cache memory prior to the cpu request. The miss rate and processor's penalty are the major issues faced by the multiprocessors, which can be reduced by the prefetching technique. This are the advantages of prefetching .Prefetching create more traffic but very less access time. This paper includes a survey of different prefetching techniques on NoC.

Keywords—prefetch; miss rate; processor penalty;

I. INTRODUCTION

In NoC, Processing units are interconnected via packet based network. Each resource is called as a 'tile' and all resources organized as rectangular tiles on the chip. All the tiles have an address. Tiles are interconnected by network of routers communication by packet transmission. Fig. 1 shows architecture of NoC. The buffer between processor and memory is cache. It is trivial but fast. Old values will be eliminated from caches to make space for new values. Anytime, the programs can access relatively small portions of their address area, as stated by the theory of locality. Whereas in temporal locality, when an item is remarked, it will tends to be remarked again soon. In spatial locality, if an item is remarked, then the items whose addresses are closer will tends to be remarked soon. Cache memories are controlled in hardware by cache controllers. It hold routinely accessed block of main memory. CPU looks for data first in L1, followed in L2, then in the prime memory.

Memory access latency is an important part of system performance. A typical technique used to tolerate this latency is prefetching. This technique predicts the memory access that will be made by the processor and fetches the data prior to the request formed by the processor. This technique is very adequate when prefetch latency and timeliness are fairly high. If the processor wants the prefetched data, then the prefetch request is effective or accurate. That means all the prefetch requests are not effective or useful. It is based on the processor. Inaccurate prefetch requests disuse the system

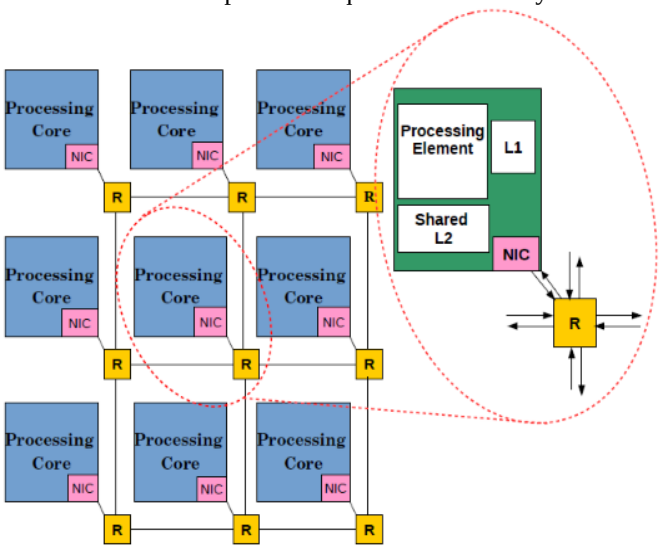

Fig. 1: Architecture of NoC

resource such as memory, cache space and decreases the performance by delaying demand or useful requests. Memory access time is based on the distance between the requesting processor and memory storing the data and also on the network traffic. The data prefetching leads to reduce the penalty in NoC based multiprocessor. The time to predict data is based on cache miss history of each processor and NoC's traffic information. A useful prefetching technique must predict what data to prefetch and when to prefetch. If right data is successfully loaded into the cache on time to processors use, system performance maximized. There are many techniques used to decide when to prefetch data. Most of them are triggered based on the existence of cache miss or when accessing the prefetched data. The on chip interconnects /network (NoC) is a demanding shared resource in multicore system. This is the communication link that connects the core and caches carrying private cache miss traffic and coherence

traffic. Miss predicted prefetch requests disuse NoC resources and delays other requests while traversing the NoC.

STREAM BASED PREFETCHING

In 2013 Jamie Garside, Neil C. Audsley presented a paper titled “Prefetching across a shared memory tree within a Network-on-Chip architecture”[4] proposes, Stream based prefetching unit take advantages of a separate shared memory tree, that will provide access to external memory from each cpu tile. Stream prefetching makes a simple premise that if the memory blocks at addresses n-2, n-1 and n have been fetched in sequence, then it is likely that the processor soon require block n+1.This can be implemented by the lookup table and indexed on the last miss address for the stream corresponding to a table row. The miss address will try to find out from the table during the cache miss. On a cache hit, if the cache line is tagged as prefetched, then the prefetcher is notified, which will again lookup the address in the table, update it and will enable a request for the adjacent line. If there is no row corresponding to a cache miss address, a different row is picked as a replacement candidate, typically using LRU or round robin selection and filled in with the information of the miss address. Stream prefetching can only detect and prefetch a subset of all possible traffic patterns, but it is ideal for code prefetching.

A Markov prefetching stores the memory addresses came next in the reference stream and prefetches those addresses too. This technique demands a broad expanse of storage space to encode this table. Generalization of this technique stores the differences between the memory addresses rather than the addresses themselves.

TEMPORIZED DATA PREFETCHING

In 2016 Maria Cireno, Andre Aziz, Edna Barros presented a paper titled “Temporized Data Prefetching Algorithm for Noc-based Multiprocessor Systems”[1] Temporized data prefetching mechanism is to minimize penalty in NoC based multiprocessor. Temporized prefetching algorithm is to perform a prediction based prefetching using time prediction to decide what and when to prefetch. This system composed by two main components- client and server. For each NoC processing node, there is a prefetching system. The client is responsible of getting information about prefetching network requests and acting on cache controller to load prefetched data into local cache. The server is responsible of monitoring directory and notifying client to perform prefetching requests when predicted time is reached. Server uses the history of cache miss occurrences and time series prediction to estimate when the processor will requests new data. Data is prefetched to avoid cache miss and reduces processors penalty. Fig. 2 shows prefetching state machine.

In the prefetching state machine, the initial state waits for the cache miss occurrences. When it happens, the prioritized prefetch flow is initiated. Client prefetching send a message to server requesting missed data. When this message arrives at the server, the time of cache miss is registered. Time series prediction is based on the Holt algorithm. It is used to predict next processors request. Data address is send to multilevel difference table predictor to predict the next missing address using miss history to find an access pattern.

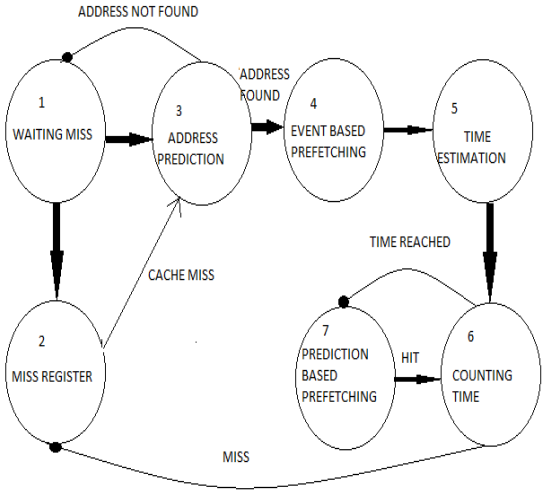

Fig. 2: Prefetching state machine

This process is repeated until the MLDT predictor finds a data address access pattern and calculates the next address. When the next missing address is predicted, the first prefetching request is performed using an event based prefetching. Holt algorithm provides the estimated time for next cache miss occurrence and the time necessary for data to travel through the NoC. These times are mainly used to determine when the prefetching request will be processed. The state machine goes through counting time process, once the prefetching time is calculated. An internal counter waits until time is reached to perform prefetching request. If prefetching is unsuccessful, then it generates a cache miss, so that state machine returns to state 2 to register cache miss and update information to adjust predictions. The counter is set to zero and counting restarts when the prefetching becomes successful.

APPLICATION-AWARE PREFETCHING

In 2012 Nachippan Chindambaram Nachippan, Asit K. Mishra presented a paper titled “Application-aware Prefetch Prioritization in On-chip Networks” [5] proposes NoC routers include the prefetching from different types of applications and prioritize prefetching from these applications. Prefetching is used to maximize the performance by the application and it will create interference to other applications. Aggressive prefetching in a NoC based system can issue

network congestion so that the prefetcher can inject a large number of inaccurate prefetches into the network. These inaccurate prefetching can delay demand requests as well as helpful prefetch requests. In memory controllers as well as caches, the prefetch and the demand requests are get prioritized, therefore imprecise prefetch requests have highest priority over demand requests. When demand request have highest priority over prefetch requests. So, it will provide a good performance than treating prefetches and demand equally. Prefetch requests from different applications have different impact on performance of the application and system performance.

Prioritize prefetches among the applications that attains maximum from prefetching and does not cause any issues to other applications. The two matrices for applications are Prefetch accuracy and prefetch count, by monitoring of which the prefetch could be approximated. An application having high prefetch accuracy attains more benefit from the prefetching. Also applications that inject fewer prefetches into network causing less issue to other applications, because there are fewer chances to interface with other application's requests. The applications with high prefetching accuracy and low prefetch count have highest priority and attains more from the prefetching and also less harmful to other applications because of the prefetching.

BALANCED PREFETCHING

In 2014 Andre Aziz, Edna Barros, Maria Cireno presented a paper titled "Balanced Prefetching Aggressiveness Controller for NoC based Multiprocessor" [2] proposes increasing the prefetching aggressiveness can reduce cache miss and can improve overall system performance. Higher levels of prefetching aggressiveness consume more network and memory bandwidth to perform prefetching transactions, which increase memory access time and processor penalty. It is needed to balance prefetching aggressiveness to minimize processor penalty. Aggressiveness control is very important in multiprocessor environment. Ebrahmi proposed the first model of global feedback to analyze the impact of interference in inter processors prefetching. Hierarchical control aggressiveness prefetcher that dynamically downgrades the local aggressiveness and it causes a negative interference on the overall system. ABS controller performs prefetching in the shared LLC. The miss rate of shared LLC reduced by adjusting the aggressiveness, this is done by ABS and it will increase system performance. Hill climbing is an approach to set prefetching more or less aggressive based on lest recently miss rate.

Balanced prefetching aggressiveness approach consists in five main steps.

- Capture
- Address prediction
- Aggressiveness prediction
- Send request
- Execute request

In the first step, capture read miss transaction that arrives at the directory. In the second step, in which to predict what is the next address demanded by the processor. Aggressiveness prediction will predict how many cache blocks should be sent to private cache. The sent request build network packet with all predicted block address. In the last step, in which make a read request in private cache of predicted addresses present in network packet.

MUTUALLY AWARE PREFETCHING

In 2014 Junghoon Lee, Hanjoon Kim, Minjong Shin presented a paper titled "Mutually aware prefetcher and on chip networks designs for multicores" [3] proposes hardware prefetching is an effective technique in the processors with higher performance. In multicore architecture, core communicating through the on-chip networks. Traffic generated by the prefetcher can account up to 60% of the total on chip network traffic. The distinct characteristics of prefetch traffic have not been considered in on chip network design. Hardware prefetcher is used to predict different data access pattern. Commonly used prefetching techniques are based on stream detection. Stream prefetcher prefetches non unit stride cache lines by dynamically tracing streams. Prefetcher will create a stream entry with limited training window once a cache miss occurs. The stream entry is trained with the address differences between the current miss address and prior addresses stored in the stream entry. The stream prefetcher will generates one or more prefetch request when a stream entry is trained. The number of prefetch request generated by the stream prefetcher is called the degree of prefetch. A stream prefetcher can detect multiple streams simultaneously.

Different techniques in prefetch aware networks are source level prioritization, multi-level TPA, dynamic priority boosting. The source level prioritization includes, this is the most simple prioritization mechanism for demand requests over prefetch request is to reorder the request at the source node. Request packet pending at the source node, if the router of the source node can no longer accept a new packet. As soon as the router has a tree buffer, source level prioritization enforces demand packet to be injected to the router even if older prefetch packet exist. This mechanism does not require any changes in the network and require only a minor change in the request selection logic of the cache controller.

Multi-level TPA is the first optimization for TPA is to support multiple priorities for prefetch traffic. Multi-level TPA divides prefetch priority in two levels. Urgent prefetch and less urgent prefetch packets. In the dynamic priority, boosting is to increase the priority of the Packet dynamically. When the slack from a prefetch issue to a demand access or

approach to the same cache line in short, the demand access may occur .While the prefetch packet is in the network or expecting for the data response from the memory. In similar cases, the prefetch packets must be returned quickly as possible. As soon as a demand miss occurs on the similar cache line as an outstanding prefetch request, then the subsequent packet serving the prefetch is upgraded to demand packet. In the network aware prefetchers, the traffic aware throttling, which detects network profusion dynamically and throttles prefetch request, if the network become congested. Crowded path from source to target is a path in network which has much longer packet latencies than the unloaded latency from the same source to target pair. A mechanism is utilized to measure the profusion status for each path by checking the time stamp in the packet. The receiving node might determine the transfer latency for each packet. If latency of the arrived packet is a bit large than the unloaded latency from where the source designate the line as a congested one. Consider a network and N is the total nodes in number, when the prefetcher generates a prefetch it checks whether the path to the end node is congested.

COMPARISON OF DIFFERENT PREFETCHING TECHNIQUES

This chapter includes comparison of different prefetching techniques. In stream prefetching [4] only detect and prefetch subset of all possible traffic patterns but it is ideal for code prefetching. The Markov prefetching stores the memory address came next in the reference stream and prefetches those addresses too. This technique demands a broad expanse of storage space to encode the table. This stores the difference between the memory address rather than the addresses themselves.

Temporized data prefetching [1] minimizes penalty in NoC based multiprocessors. It is a prediction based prefetching using time prediction to decide what and when to prefetch. This system composed by two parts client and server. Client is responsible of getting information about prefetching network requests and acting on cache controller to load prefetched data into local cache. Server is responsible of monitoring directory and notifying client to perform prefetching requests when predicted time is reached. Server uses the history of each miss occurrences and time series prediction to estimate when the processor will requests new data.

Application aware prefetching [5] includes prefetching from different types of applications and prioritizes prefetching from these applications. Prefetch requests from different applications have different impact on performance of the application and system performance. By prioritizing prefetches among the application does not cause any issues to other applications. Two matrices for applications are prefetch accuracy and prefetch count. An application having high prefetch accuracy attains most benefit from the prefetching. The application with high prefetch accuracy and low prefetch count has highest priority.

Increasing the prefetching aggressiveness [2] can reduce cache miss and can improve overall system performance. Higher levels of prefetching aggressiveness consume more network and memory bandwidth to perform prefetching transactions. Balanced prefetching aggressiveness is used to minimize processor penalty. Aggressiveness control is very important in multiprocessor environment. Hill climbing is an approach to set prefetching more or less aggressive based on least recently miss rate.

Different techniques in prefetch aware networks [3] are source level prioritization, multi-level TPA, dynamic priority boosting. Source level prioritization is a simple prioritization mechanism for demand requests over prefetch requests. Demand packet to be injected to the router, even if older prefetch packet exist. This mechanism requires only a minor change in the request selection logic of the cache controller. Multi-level TPA is the first optimization of TPA and supports multiple priorities for prefetch traffic. This divides prefetch priority in to two level urgent prefetch packets and less urgent preftch packets. In dynamic priority boosting, increase the priority of the packet dynamically. Prefetch packets returned quickly as possible. Traffic aware throttling which detects network profusion dynamically. Throttles prefetch request, if the network become congested. A mechanism is utilized to measure the profusion status for each path by checking the time stamp in the packet.

Stream Prefetching	Temporized data Prefetching	Application Aware Prefetching	Prefetching Aggressiveness	Mutually aware prefetching
Only detect and prefetch subset of all possible traffic patterns.	Prediction based prefetching.	Prefetching from different types of applications and prioritizes prefetching from these applications.	Reduce cache miss.	Aware about both prefetch aware networks and network aware prefeters.
Markov prefetching is based on stream prefetching.	Minimizes penalty.	By prioritizing prefetches among the applications does not cause any issues to other applications.	Improve overall system performance.	Different techniques in prefetch aware networks are source level prioritization, multilevel-TPA, dynamic priority boosting.
Stores the memory asddresses came next in the reference stream and prefetch those addresses too.	Use time prediction to decide what and when to prefetch.	Applications with high prefetch accuracy and prefetch count have highest priority.	Consume more network and memory bandwidth.	Prefetch packet returned quickly.
Demands broad expanse of storage space.	Composed of two main components – client and server.	prefetch count and prefetch accuracy are the two matrices which improves performance..	Minimizes processor penalty.	In the network aware prefetchers, the traffic aware throttling which defect network profusion dynamically.

Table. 1: Comparison of different prefetching techniques

II. CONCLUSION

Prefetching in a system based on NoC significantly increases system performance. It will help to decrease the processor penalty and miss rates. Prioritizing demand packets over the prefetch packets at the source node or at the routers resulted in significant performance improvements. As compared to other prefetching techniques, mutually aware prefetcher is more efficient. Stream prefetching only detect and prefetch subset of all possible traffic patterns. Whereas, temporized data prefetching is a prediction based prefetching. In application aware prefetching, the prefetching is from different types of applications and prioritizes prefetching

among these applications. But mutually aware prefetching is different from all of the above prefetching techniques. It mainly focuses on prefetch aware networks and network aware prefetchers. It also includes different prioritizing mechanisms which help to ease the prefetching. So mutually aware prefetching is more effective. Multicore processors have turn into a necessary part for future digital technologies of NoC. Efficient NoC designs are the backbone for future kilo-core chips.

REFERENCES

[1] Maria Cireno, Andre Aziz, Edna Barros, "Temporized data prefetching algorithm for NoC based multiprocessor systems," IEEE 27th Inter national Conference on pp. 235-236, December. 2016.

[2] Andre Aziz, Maria Cireno, Edna Barros, "Balanced Prefetching Aggressiveness Controller for NoC-based Multiprocessor," 27th Symposium on Integrated Circuits and System Design, Sept. 2014.

[3] Junghoon Lee, Minjong Shin, Hanjoon Kim, "Mutually Aware prefetcher and on-Chip Network Designs for Multi-cores ," IEEE Transaction on Computers vol. 63, pp. 2316-2329, Sept. 2014.

[4] Jamie Garside, Neil C. Audsley, "Prefetching across a shared memory tree within a Network-on-Chip architecture,"International Symposium on System on Chip, December. 2013.

[5] Nachippan Chindambaram Nachippan, Asit K. Mshra, "Application aware prefetch prioritization in on chip networks," 21st International Conference on Parallel Architectures and Compilation Techniques, Sept. 2012.