A Non-Uniform Cache Architecture on Networks-on-Chip: A Fully Associative Approach with Pre-Promotion

Akio Kodama
Systems Research & Development Center
NS Solutions Corporation

Toshinori Sato
Department of Artificial Itelligence
Kyushu Institute of Technology

Abstract

Global interconnect becomes the delay bottleneck in microprocessor designs, and latency for large on-chip caches will be intolerable in deep submicron technologies. The recently-proposed Non-Uniform Cache Architectures (NUCAs) exploit the variation in access time across subarrays to reduce typical latency. In the dynamic NUCA (D-NUCA) design, a set-associative structure is selected and thus the flexibility of data placement and replacement is limited. This paper investigates one of the unexplored design space; a fully associative approach. In addition, we propose a pre-promotion technique to reduce the number of incremental search in the distributed cache banks. We show that, compared with a traditional multi-level cache, up to 110% improvement in IPC is achieved at 30nm.

1 Introduction

While the advances of semiconductor technology enables large scale systems, it presents a serious issue on large on-chip caches for microprocessors, accompanied with deep pipelining. Modern microprocessors heavily rely on on-chip caches to keep their high-performance. As capacity of on-chip caches increases, the length of wires connecting the caches with processor is also increasing. Wire delay will limit the ability of microarchitectures to improve instruction throughput [1]. While delay of transistors is improved as their feature sizes shrink, delay of wires, especially of global wires, is growing, and thus discrepancy between wire and transistor delays makes it a major issue to manage wire delay in nanometer scale designs.

Kim et al. [11] recently proposed Non-Uniform Cache Architectures (NUCAs) which exploit the variation in access time across subarrays to reduce typical latency. While performance has been analyzed for various cache organizations, the flexibility in data placement is limited due to adopting the set-associative mechanism for mapping. As mentioned in [11], selecting fully associative approach might cause serious overhead of locating lines in cache. To the best of our knowledge, the most flexible approach has not been evaluated. In this paper, we explore the extreme for providing each cache line with maximum flexibility. Furthermore, we propose a new promotion policy to reduce the time required for searching requested lines.

The organization of the rest of this paper is as follows: Section 2 introduces the background of our work. Section 3 presents an alternative design for NUCAs. Section 4 explains our evaluation methodology and Section 5 presents experimental results and discussion on the effectiveness of our approach. Finally, Section 6 concludes the paper.

2 Background

2.1 Wire Delay

Improvement in semiconductor technology increases processor performance by 50–60% per year [1]. However, currently, we are facing a big problem that the delay of wire relative to transistors are increasing. The wire delay is given by

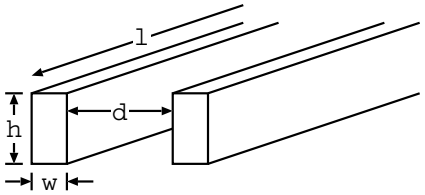


Figure 1: Geometry of wires

$$D_{wire} \propto R_{wire} \times Cwire \tag{1}$$

$$R_{wire} \propto \frac{l}{w \times h} \tag{2}$$

$$C_{wire} \propto \frac{h \times l}{d} \tag{3}$$

where l, w, and h are the length, the width, and the height of the wire, and d is the distance between two wires, as shown in Figure 1. While the width and the distance are reduced as semiconductor technology is improved, the hight is tend to be kept constant and the length of global wire are not reduced but are rather increased. Thus, global wire delay becomes a serious problem in nanometer scale designs.

2.2 Globally Asynchronous Locally Synchronous Architectures

Deeper pipelining combined with growing wire delay limits the chip area that is reachable in a single clock cycle. Thus, clock distribution on modern microprocessors is becoming more challenging. In order to solve the problem, Globally Asynchronous Locally Synchronous (GALS) architectures have interested designers [9]. In GALS, the system is partitioned into several synchronous blocks which asynchronously communicate with each other. One of the key trend of increasing interests in GALS is that it offers a good compromise between single-clock synchronous design and purely asynchronous design. In GALS design, there are fundamental difficulties of interfacing systems operating under different timing and of handling long communication delays between the systems. They are managed by using asynchronous FIFOs [4].

2.3 Networks-on-Chip

In addition to the global wire delay problem, nanometer scale design is facing the other challenging interconnect issues of signal integrity and energy consumption. Considering these, next generation Systems-on-Chip (SoC) designs will use scalable and reliable point-to-point transformation [7, 8, 10]. This Networks-on-Chip (NoC) architecture is a data packet based communication network on a single chip. A lot of blocks with standardized size are connected by a switched network. Switches route and buffer

packets between blocks. Each switch is connected with each other through input and output channels, and may have internal FIFOs to handle congestion. The structured wires manage global delays and provide efficient global communication.

2.4 Caches

Modern microprocessors have large on-chip caches. IBM's POWER5 has 1.92 MB L2 cache, Fujitsu's SPARC64V has 2MB L2 cache, and Intel's Montecito will contain 24 MB of on-chip L3 cache. These large on-chip caches will require large access latency, because global wire delay is increasing in nanometer scale technologies as mentioned above, resulting in poor processor performance.

Recently, Kim et al. [11] proposed NUCAs to solve the problem. NUCAs exploit the variation in access time across subarrays to reduce typical latency, and thus they allow fast access to subarrays close to processor while retaining slow access to far subarrays. In addition, dynamic NUCA (D-NUCA) adopts network-based design and exploit locality by promoting frequently accessed data to fast and close subarrays from slow and far subarrays, resulting in the decrease in the average access latency.

Cowell et al. [6] extended D-NUCA and verified it using a realistic floorplan, cache sizes, and wire delays of HP's 21364. Furthermore, they evaluated additional topologies for the network and improved the policy for the promotion.

Based on the NUCA concept, Chishti et al. [5] proposed Nonuniform access with Replacement And Placement usIng Distance associativity (NuRAPID) to solve some problems in NUCAs. NuRAPID decouples tag and data arrays and improves the flexibility of data placement and the energy efficiency.

An alternative approach to manage large on-chip caches is utilizing on-chip transmission line technologies. Transmission Line Caches (TLC) proposed by Bechmann et al. [3] is such an approach, and improves the area and energy efficiencies.

3 Network-based Adaptable Cache

In conventional cache memories, their hit latency is uniform, and thus it is determined by the longest access latency to the furthest bank. Future cache memories are requested a performance scalable with improving semiconductor technologies. To satisfy the requisite, we should remove long wires for diminishing serious impact of wire delay, and should improve locality for reducing an average access latency.

3.1 Overview

We propose to realize a large cache on the NoC architecture. It consists of a lot of small banks, each of which is communicate with each other using a communication network. Figure 2 shows the network-based cache. Each bank has a full-associative memory (denoted as `CAM`) and is connected with an intelligent router (denoted as `IR`). It also has its dedicated clock; that means the network-based cache is implemented in GALS style. As mentioned above, the NoC architecture eliminates long wires with indeterminate delay. The IR transfers data requested by the processor using a packet-based communication. Since each cache node works independently with its dedicated clock, there is an asynchronously FIFO between two nodes to solve the problems mentioned in Section 2.2. In this scheme, when a hit occurs to a far bank, the requested cache line is promoted to the closest bank in parallel with its movement. In addition, the IR perform pre-promotion to improve locality. It predicts cache lines which will be requested and transfers them to close banks from far banks. This might reduce an average access latency. To realize the flexibility in locating the cache line, we select the full-associative memory as each bank. To the best of our knowledge, this flexible approach has not been evaluated. In the rest of this paper, we call this network-based

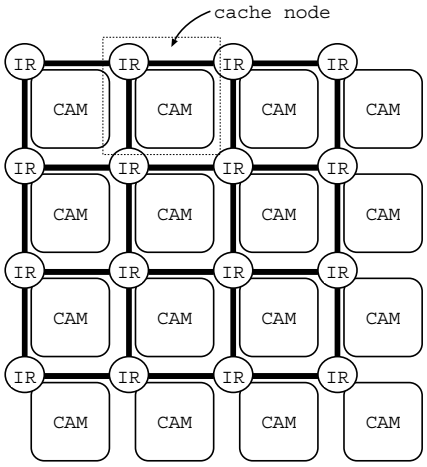


Figure 2: Network-based adaptable cache

adaptable cache *full-associative NUCA* (FA-NUCA). Tag arrays are distributed and thus multicast is required for finding data requested by the processor.

3.2 Read Transaction

As Figure 3 shows, read transaction is conducted as follows.

1. A control packet is generated by the processor to request a cache line and is multicasted to whole network including memory controller, which manages the main memory or L3 cache. This packet includes the address of the requested line. (Figure 3_i)

2. The cache node that has the requested cache line injects the line into the network as a data packet and the packet is delivered to the processor. We call the node *owner node*. In addition, the closest cache node will capture the requested line. (Figure 3_i)

3. In parallel with the step 2, a control packet is generated by the owner node to cancel the lower level memory access. (Figure 3_ii)

4. When the requested line gets to the closest node, a control packet is generated by the closest node to announce that it has captured the line, and the packet is delivered to the owner node to invalidate the line kept in the owner node. (Figure 3_ii)

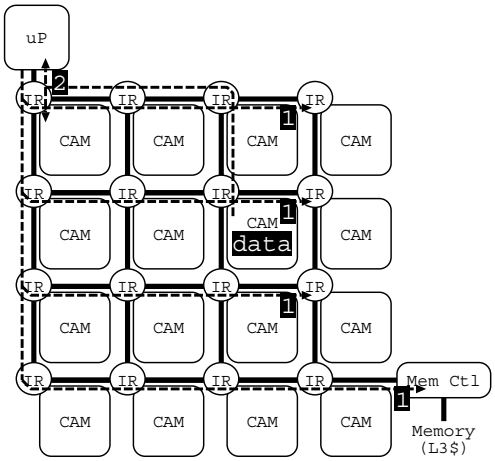
5. When the closest node is full, the LRU cache line is evicted to the next node and the line swapping continues by as packet transforming until an empty line is found. If there is no empty line in FA-NUCA, the latest evicted line is replaced in the lower level memory. (Figure 3_ii)

3.3 Write Transaction

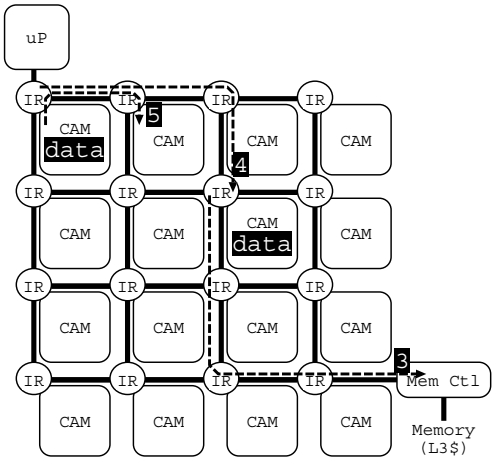
As Figure 4 shows, write transaction is very similar with the read transaction.

1. A control packet is generated by the processor to invalidate the stored date kept in its owner node. (Figure 4_i)

2. The data packet is injected into the closest node, and the stored data is kept in the bank. (Figure 4_i)

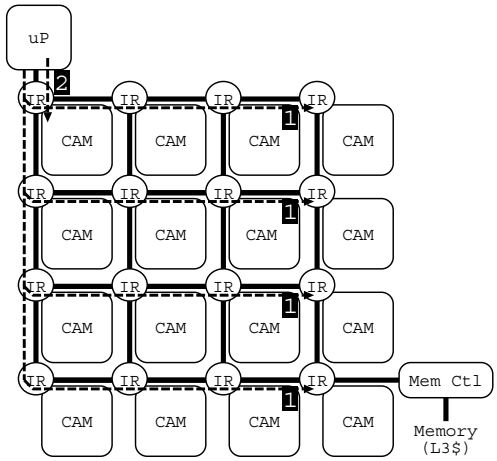


(i)

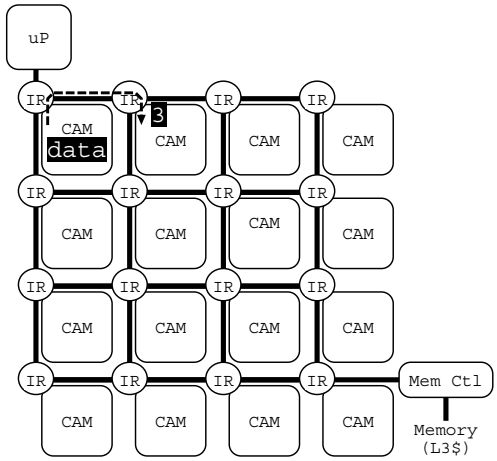


(ii)

Figure 3: Read transaction



(i)



(ii)

Figure 4: Write transaction

3. When the closest node is full, the LRU cache line is evicted to the next node and the data swapping continues until an empty line is found. If there is no empty line in the cache, the latest evicted line is replaced in the lower level memory. (Figure 4_ii)

4 Evaluation Methodology

We implemented our simulator using MASE/ SimpleScalar/ Alpha tool set (version 4.0) [12], and use SPEC2000 benchmark for our evaluations. Table 1 lists the programs and their input data sets. We use the object files provided by the University of Michigan. They were compiled by DEC C V5.9-008 on Digital UNIX V4.0 (Rev.1229). For each program, 1 billion instructions are skipped before the actual simulation begins and the next 200 million instructions are executed. We do not count nop instructions.

We perform our simulations for 70nm and 30nm technologies, with clock frequencies of 5 GHz and 10 GHz, respectively [1]. Table 2 shows the configuration for the simulated processor, which is commonly used at both 70nm and 30nm. Table 3 shows the base configurations for the conventional multi-level uniform cache architecture (ML-UCA) L2 cache, and Table 4 shows the base configurations for the FA-NUCA L2 cache. The L2 cache size is determined following [11]. To estimate the cache access latency, we used Cacti (version 3.2) [13]. Because it is not expected that memory latency is considerably improved, the memory latency at 30 nm with a clock frequency of 5 GHz is determined as two times longer than that at 70 nm with 10 GHz. The bank size and the number of banks in the FA-NUCA are determined based on the following considerations. The miss penalty of the FA-NUCA includes latency for traversing the network between processor and memory controller, which place at the both ends of the network diagonally. We can calculate it by multiplying the latency to move a packet one hop by the Manhattan distance of the network. As explained later, the former is 1 cycle and thus the number of banks can be determined. After that, we determine the bank size by dividing the total cache size, which equals to that of the ML-UMA, by the number of banks.

A 2D-mesh topology is selected for the simulated interconnection network. Considering other topologies [6] is remained for the future study. We use a minimal adaptive routing strategy [2] considering loads at possible destination banks; that means that packet routes are adapted based on whether input FIFOs in the destination banks have enough space. During our simulations, we have not detected any deadlocks. The capacity of each FIFO is 10 packets. In the switch, packet transfer requires one cycle including routing and transferring.

Table 1: Benchmark programs

program	input set
164.gzip	ref.in
176.gcc	expr.i
181.mcf	inp.in
197.parser	ref.in
255.vortex	lendian1.raw
172.mgrid	mgrid,in
177.mesa	mesa.in
183.equake	inp.in

Table 2: System parameters

L1 I-cache	ideal
Br predictor	2K-entry bimode
Fetch width	4 instructions
Issue width	4 instructions
Commit width	4 instructions
RUU size	16 instructions
LSQ size	4 instructions
L1 D-cache	8 KB, 4-way, 64 byte block, 2 cycle hit, 1 port, pipelined

Table 3: ML-UCA L2$ parameters

	70 nm	30 nm
Size	8 MB	32 MB
Access latency	36 cycles	74 cycles
Block size	64 bytes	
Associativity	8-way	
Memory latency	132 cycles	264 cycles

Table 4: FA-NUCA L2$ parameters

	70 nm	30 nm
Bank size	128 KB	128 KB
Bank access latency	6 cycles	5 cycles
# of banks	8 x 8	16 x 16
Total size	8 MB	32 MB
Block size	64 bytes	
Memory latency	132 cycles	264 cycles

About pre-promotion, we borrow one block lookahead (OBL) strategy [14] from prefetching techniques. If line $\#i$ is referenced, only line $\#(i+1)$ is considered for pre-promotion. Lines $\#i$ and $\#(i+1)$ do not have to be in the same bank. During traversing a packet requiring line $\#i$ to its owner bank, if an intermediate bank find that it has line $\#(i+1)$, it promotes the line to processor. However, if the line $\#(i+1)$ is held in a bank farther than line $\#i$'s owner bank, no pre-promotion is triggered because the request packet disapears in the owner bank.

5 Simulation Results

In Section 5.1, we show that processor performance is improved by up to 40.4% at 70 nm, when the conventional ML-UCA is replaced by the FA-NUCA. We use instructions per cycles (IPC) as a metric for evaluating processor performance. Section 5.2 demonstrates that the FA-NUCA has a technology scalability and achieves up to 108.9% improvement in processor performance at 30 nm.

Table 5: L1 data cache miss rate

program	%miss rate
164.gzip	3.57
176.gcc	2.68
181.mcf	0.13
197.parser	4.05
255.vortex	3.66
172.mgrid	16.0
177.mesa	1.98
183.equake	0.06

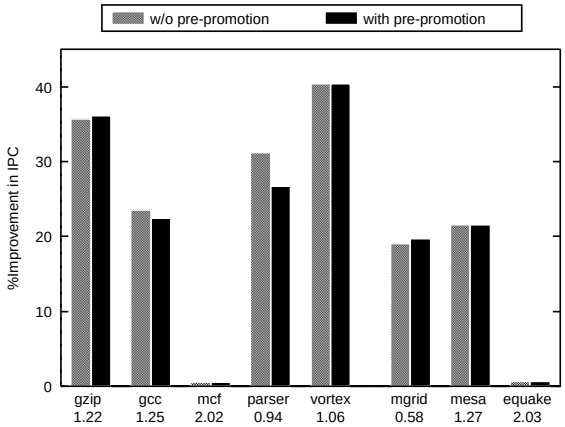


Figure 5: %Improvement in IPC at 70 nm

5.1 Results at 70 nm

Table 5 shows miss rates in L1 data cache. We can find considerable miss rate for most programs except for `181.mcf` and `183.equake`. We expect improvement in processor performance in these programs. Figure 5 shows the percent improvement in processor performance when the conventional ML-UCA is replaced by the FA-NUCA. For each group of two bars, the left bar is for the case where pre-promotion is not utilized and the right one is for the case where pre-promotion is turned on. Numbers under program names are the baseline IPCs.

Most programs benefit significantly from replacing the ML-UCA by the FA-NUCA. As we expected, programs with larger miss rate (`164.gzip`, `197.parser`, and `255.vortex`) achieves larger IPC gain in general. Especially `255.vortex` achieves 40.4% higher IPC with the FA-NUCA than with the ML-UCA. In contrast, `172.mgrid` has a marginal performance improvement instead of its largest miss rate. This is because as follows. Figure 6 shows instruction mix of memory operations. The lower part indicates the percentage of load instructions and the upper one does that of stores. As we can observe, `255.vortex` has the smallest percentage of load instructions and `172.mgrid` has the largest one. A load instruction has more packet traffic than a store does, and thus `255.vortex` gains most and `172.mgrid` gains least. On the other hand, `181.mcf` and `183.equake` benefit little. This is due to their small L1 data cache miss rates.

Refering back to Figure 5, we can not find substantial performance gain with pre-promotion. Instead, we find performance degradation for some programs (`176.gcc` and `197.parser`). One of the reason is that pre-promotion increases packet trafic in the network and thus L2 cache latency is increased. Another reason is that unexpected demotions occur due to the limited capacity of the closest bank. Figure 7 shows the distribution of distance between

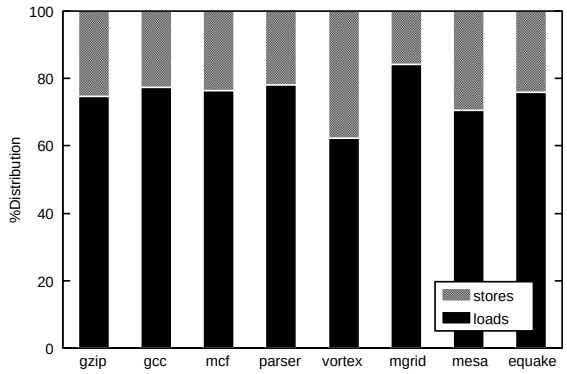


Figure 6: %Breakdown of memory operations

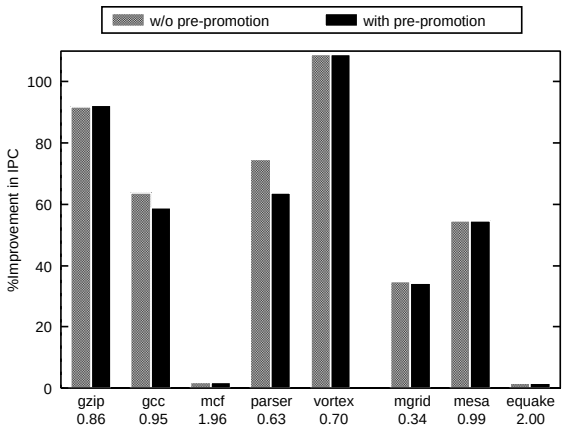


Figure 8: %Improvement in IPC at 30 nm

processor and cache banks where requested data are found. In almost all programs except `177.mesa`, more data an be found in the closest bank. However, In the cases of `176.gcc` and `197.parser`, where performance degradation is observed with pre-promotion, hits at more distant banks are increased. This means that unexpected demotions occur due to useless pre-promotions. In other words, improvement in pre-promotion algorithm is required for filtering useless pre-promotions.

Findings from simulations so far are summarized as follows. Utilizing the FA-NUCA results in an average of 21.6% improvement in IPC over the conventional ML-UCA approach. Overload in the network, especially due to load instructions, limits the benefit obtained from the FA-NUCA. Pre-promotion improves the utilization of the closest bank; that means more accesses hit at the closest bank. However, any performance gain is not obtained due to the increase in pachet trafic and the unexpected demotions.

5.2 Results at 30 nm

Figure 8 shows performance gain with the FA-NUCA over the ML-UCA at 30 nm. The layout of Figure 8 is the same as for Figure 5. It is easily observed that the baseline IPCs are significantly diminished. This means that L2 cache latency is one of the dominant components that determine processor performance. Due to the decrease in the baseline IPCs, all programs obtain more benefits from the replacement of L2 cache at 30 nm than at 70 nm. Performance gain is an average of 50.0% and up to 108.9% in `255.vortex`. This means that FA-NUCA has better scalability than the ML-UCA does. While absolute values are different from the results at 70 nm, the tendencies observed are almost same between at 30 nm and at 70 nm. The explanations why are same with the discussions in the previous section. Similarly, we find little IPC gain via pre-promotion. Due to the limit in space, we do not show the distribution of distance between processor and servicing cache banks at 30 nm. However, its tendency is also equivalent between at 30 nm and at 70 nm.

6 Conclusions

In this paper, we explored an alternative NUCA design, FA-NUCA. It has a merit of large flexibility in placement but has possible demerits of the delay for searching requested data and of large packet traffic. Based on detailed simulations, we found that in most cases FA-NUCA significantly improves processor performance compared with the conventional ML-UCA. We showed that up to 110% improvement in IPC is achieved at 30nm. In addition, we proposed a pre-promotion technique to reduce the number of incremental search in the distributed cache banks. Unfortunately, it has not been justify the increased complexity to implement the pre-promotion because it has almost no contribution on performance improvement.

One of the future directions regarding this study is evaluating FA-NUCA on the platform of chip multi-processors (CMPs). As mentioned in [11], CMPs will benefit from NUCAs. However, it is not clear how to promote frequently used data on a CMP with set-associative NUCA. FA-NUCA, on the other hand, has a maximum of flexibility in placement, and thus solve the difficulty to promote multiple data independently to different processors. One of the difficult issues to using FA-NUCA with CMPs is how to manage multiple requests from diffrent processors to a single datum. During thedatum is traversing in the network, false miss hit might occur and it might be duplicated, causing coherence problem. The other future studies are on reducing packet trafic and on improving pre-promotion algorithm.

Acknowledgments

This work was supported in part by the Intellectual Cluster program of Ministry of Education, Culture, Sports & Technology (MEXT) in Kitakyushu, and is supported in part by the Grants-in-Aid for Scientific Research from MEXT (No.15650010).

References

[1] V.Agarwal et al., “Clock rate versus IPC: the end of the road for conventional microarchitectures”, ISCA-27, 2000.

[2] H. Amano, “Parallel computers”, Sohkodo Co., Ltd., 1996.

[3] B.Beckmann et al., “TLC: transmission line cache”, MICRO-36, 2003.

[4] T.Chelcea et al., “Robust interface for mixed-timing systems with application to latency-insensitive protocols”, DAC-38, 2001.

[5] Z.Chishti et al., “Distance associativity for high-performance energy-efficient non-uniform cache architecture”, MICRO-36, 2003.

[6] C.Cowell et al., “Improved modeling and data-migration for dynamic non-uniform cache access”, 2nd Workshop on Duplicating, Deconstructing, and Debunking, 2003.

[7] W.J Dally et al., “Route packets, not wires: on-chip interconnection networks”, DAC-38, 2001.

[8] G.De Micheli, “Hot topic – network on a chip”, DATE-5, 2002.

[9] A.Iyer et al., “Power-performance evaluation of globally asynchronous, locally synchronous processors”, ISCA-29, 2002.

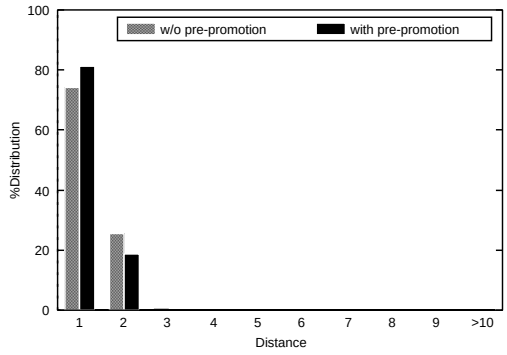
[10] A.Jantsch, H.Tenhunen (Eds.), “Networks on chip”, Kluwer Academic Publishers, 2003.

[11] C.Kim et al., “An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches”, ASPLOS-10, 2002.

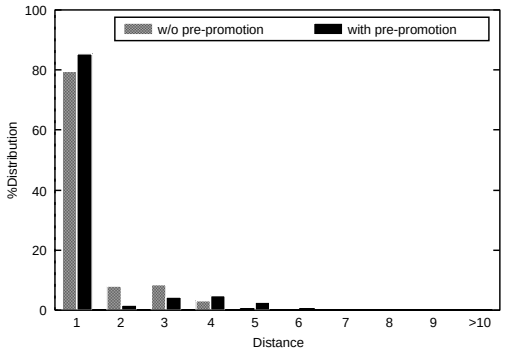
[12] E.Larson et al., “MASE: A novel infrastructure for detailed microarchitectural modeling”, ISPASS, 2001.

[13] P.Shivakumar et al., “CACTI 3.0: an integrated cache timing, power, and area model”, Western Research Lab., Report-2001/2, 2001.

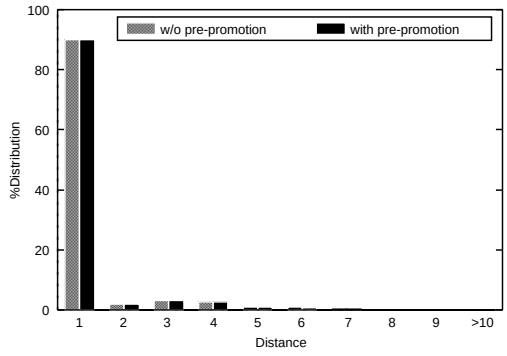
[14] A.J.Smith, “Cache memories”, ACM Computing Surveys, 14(3), 1982.



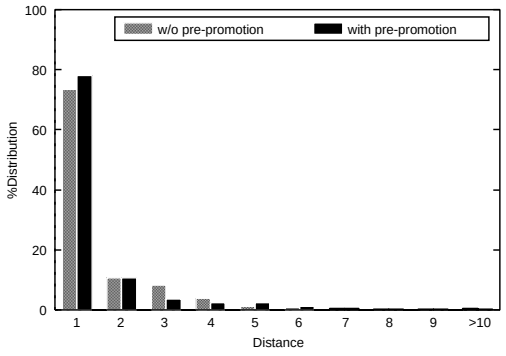
(i) gzip



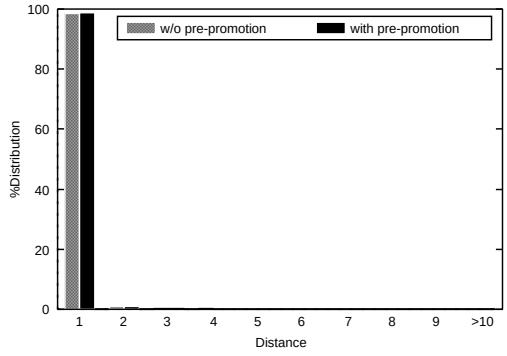
(ii) gcc



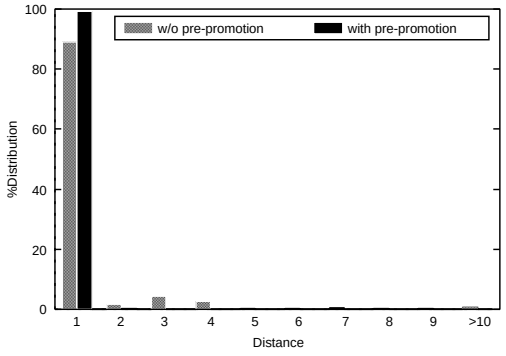
(iii) mcf



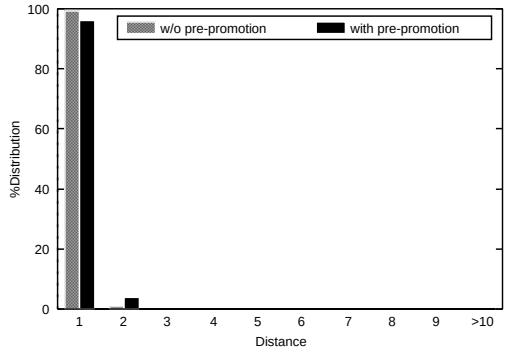
(iv) parser



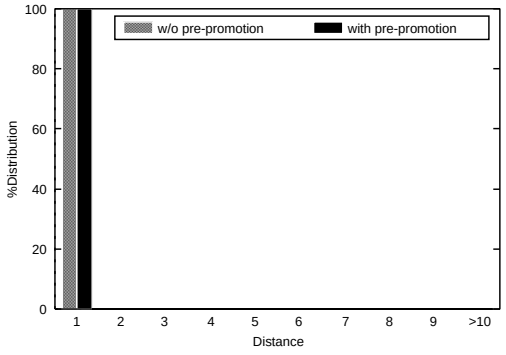
(v) vortex



(vi) mgrid



(vii) mesa



(viii) equake

Figure 7: %Distribution in distance at 70 nm