Survey on Non-Uniform Cache Architecture

Marco Antonio Zanata Alves, Philippe Olivier Alexandre Navaux
Informatics Institute
Universidade Federal do Rio Grande do Sul
Porto Alegre - RS - Brazil
{mazalves,navaux}@inf.ufrgs.br

Abstract

The non-uniform cache architecture (NUCA) is a new approach that is being discussed for the future many-core processors. These NUCA memories are an alternative to increase the performance of processors that have a very large amount of processing cores and requires a lot of data bandwidth. In the case of shared memory, one can point the access latency as the main problem for the nowadays uniform cache architectures. The NUCA memories can reduce the latency of data access, increasing scalability, thus, these memories have a great potential for increasing performance on multi-core and many-core processors. This paper presents a survey on NUCA research, presenting some research opportunities on this topic.

1. Introduction

The constant increase on the cache memory size, has leaded the cache internal to an organization with vectors of data and tags divided into several banks. The bank layout frequently used is called H-tree [5]. The area occupied by these uniform cache architecture is calculated considering the banks size and the area occupied by interconnections. Figure 1 shows an illustration of 16 banks cache memory where the lines represent communication linking all banks.

For new integration technologies the cache memories size tends to keep growing as the demand for more data throughput. Even with the cache divided into several data vectors, all these banks should behave as a one large block where the data access time usually is equal to the access time of the farther data vector. Therefore, the cache memories should be designed considering problems of latency, bandwidth, interconnection, and others.

However, for future integration technology, the cache memory using the current architecture model will suffer from wire-delay problems, so the data access time for large cache memory banks will be very slow, generating a bottleneck on the processor [2].

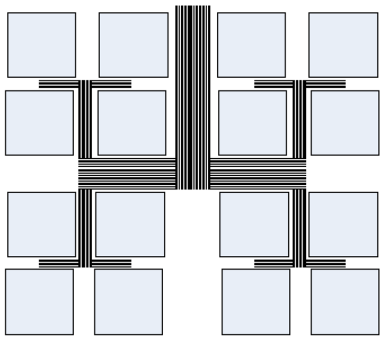
Figure 1. Organizational structure of cache memory sub-blocks [5].

This paper focus a survey on non-uniform cache architecture presenting the main architectures and techniques for these new cache memories.

This paper is organized as follows, the Section 2 presents the basic concept of NUCA memories, the Section 3 present some basic types of proposed NUCA, the Section 4 shows some static and dynamic mapping policies, the Section 5 focus on data fetch policies, the Section 6 presents some migration policies for dynamic NUCA, and finally the Section 7 present some conclusions and future work on NUCA research.

2. Non-Uniform Cache Architectures

Nowadays, multiple levels of cache memories are organized in a few discrete levels. Typically, each lower level includes and replicate the content of above levels, thereby reducing access time to levels closer to the main memory.

For future technologies, large memory cache within the chip, with a single data access time is not appropriate, since

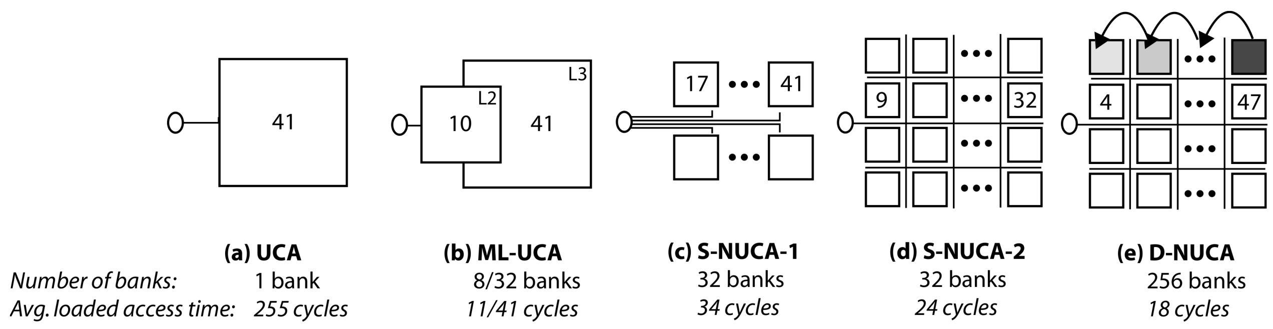

Figure 2. Several cache proposals: (a) and (b) UCA memories, (c) and (d) static NUCA memories, (e) dynamic NUCA memories. Where the data access time in clock cycles are described inside each cache bank [4].

problems related to the wire-delay inside the chip are increasing. This is the main argument for creation of new non-uniform cache architecture.

Besides the wire-delay problem, we can cite the important behavior of possible components failures along time, that next generation processors shall have [1]. Thus, some isolation or redundancy on the memory modules and interconnection can ensure that if any failure occurs the other components will keep working.

The main concept involving these new non-uniform cache architectures is that some data will reside closer to the processor, thus, these data will have some fast access times. This way, near data will be accessed faster than data which are distant from the processor. Moreover, problems associated with the number of ports and hardware failure can be partly solved by a non-centralized architecture.

3. Basic Types of NUCA Architectures

On basic operation using NUCA memories, each cache memory sub-bank should return its result as soon as possible without wait for other banks results. Concerning this basic operation, several architecture models can be proposed, the first proposal of NUCA memories [4] presented three basic types of non-uniform cache architectures, as shown in Figure 2, which compares three NUCA models with two UCA models.

Two basic models of cache memory with uniform architecture most used currently have one or two levels, which are presented in Figure 2 (a) and Figure 2 (b) respectively. One can see the gain using two levels in the cache memory hierarchy considering the data access time, given in clock cycles, illustrated in the center of each cache block.

Concerning non-uniform cache architecture two static models have been proposed. The first model has a division in sub-blocks quite similar to the internal division in the UCA memory, without the delay of waiting for response from all sub-blocks (Figure 2 (c)). Figure 2 (d), shows a model with a different type of interconnection, using a network-on-chip (NoC) to interconnect the cache blocks and the processor.

A model of dynamic NUCA memory is shown in Figure 2 (e). This proposal, uses the concept of block migration in order to increase the data access speed, to those data blocks that are used repeatedly. The block migration policy must be able to recognize the most referenced block and thus spread this data on the blocks near the processor.

4. Data Mapping in NUCA Memories

The data mapping policy in NUCA memory is what defines how the data will be mapped into banks and in which banks the data can reside. Different mapping policies can be planned, since there is flexibility for bank use in the NUCA memory [4].

Even with the flexibility given by the existence of several memory banks, the static NUCA memories (S-NUCA) provides only one static mapping for each address. Therefore, as in an UCA memory, each sub-bank is associated with several addresses. This strategy still loses much of the advantages of a NUCA memory, once that, depending on the static mapping used, some data will always have a high access time creating thus, a performance bottleneck.

For dynamic NUCA memories (D-NUCA), full flexibility is planned, where each data can be mapped into any NUCA bank. However, it must consider that the data search effort will be greater, since all banks must be searched, using some policy as directory tags or address broadcast to all banks.

An intermediate solution for the data mapping is called spread mapping [4], which uses a set of several memory

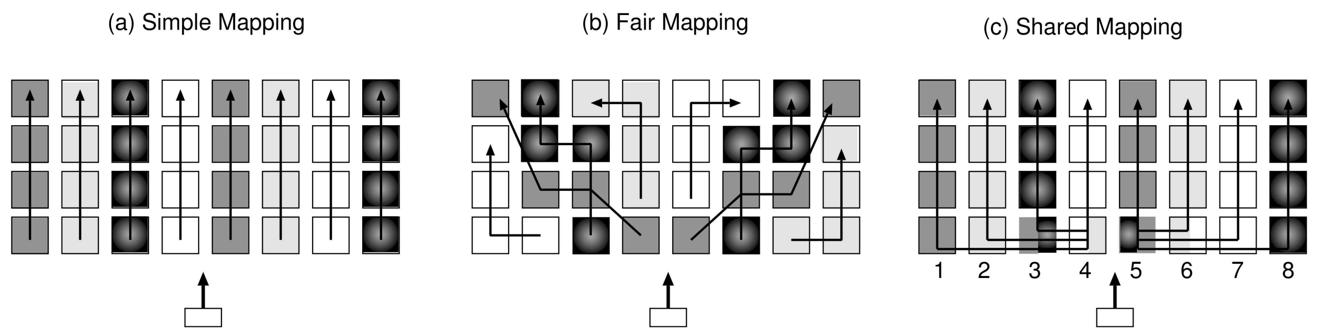

Figure 3. Different dynamic blocks allocation politics for NUCA memories: (a) simple mapping, (b) balanced mapping and (c) shared mapping [4].

banks being a set associative. Thus, each associative way will be spread over a set of several banks. Using this mapping solution, three basic allocation policy were proposed, these policies are shown in Figure 3, illustrating a 4-way set associative NUCA memory, where each set is indicated by arrows, and the external block is the processor.

On the simple mapping shown in Figure 3 (a), each column represents a set of banks and thus, each column contains one way of the set associative. To perform data search on this mapping, first it is necessary to select the row of seats (together), then select the way within the column, and then search for the tag. The disadvantages of this model is that sometimes the number of lines does not match the number of ways from the set associate, moreover, the memory access latency is not equal to all interconnect routes from the ways of set associative.

The mapping illustrated in Figure 3 (b), uses the fair mapping policy (balanced mapping), this policy addresses two problems presented by the simple mapping, with the drawback of additional routing complexity. This policy attempts to equalize the access time among the various ways of the set associative and this is the main point of this policy.

The policy of shared mapping, shown in Figure 3 (c), tries to provide fast access to all ways of set associative, sharing banks for several ways of associativity. To make this policy works, the banks close to the processor must be shared by all ways from the set, thus reducing the number of addresses in these banks close to the processor, but reducing the data access from all the ways from the set associative.

In this same context of associativity on NUCA memory, [3] propose techniques to predict the use of ways on the set associative in order to turn off ways that tend not to be used during the execution of certain applications. Creating a dynamic memory with reduced power consumption without major consequences to the final performance.

5. Data Fetch on NUCA Memories

The fetch policy defines how the data will be searched in all banks of the NUCA memory. There are two main different policies that can be used for data search on the NUCA memory banks [4].

The first policy is called incremental search, in this policy the search is performed on bank by bank until the block is found or a failure is returned by the last bank. This policy tends to reduce the number of messages on the cache interconnections and maintains low power consumption under the penalty of the system performance reduction.

The second policy is called multi-cast search, in this search, the required address is sent to all banks at once. Thus, the search is done all in parallel unless the differences by the message routing time in the interconnection. This policy tends to gain in performance with the drawback of increasing the power consumption.

Besides these two main policies, there are some alternatives such as hybrid limited multi-cast (limited multiple destinations), where the banks form sets are searched sequentially. However, the banks in each set are searched in parallel. On the other policy, called hybrid partitioned multi-cast, the banks are divided into groups that are searched in parallel, and the banks within each set are searched sequentially. In this second case, the parallelism between the banks is similar to the multiple levels UCA memories.

6. Data Migration on NUCA Memories

The data migration policy for dynamic NUCA memories, presents the conditions on which there should be a data block migration from one bank to another. The goal of data migration that occurs in dynamic NUCA memories, is to maximize the hit number of data search in banks near to the processor [4]. An interesting policy would be the LRU (Least Recently Used) sort among the lines, on the banks closest to the processor loading MRU (Most Recently Used) lines. The problem of this policy is that it could lead to large data movements, so policies must also combine power consumption and the contention increase on the data migration.

A solution of problems related to the general migration is the policy known as generational promotion. With this policy, on hit in data search, the block is sent to the processor at the same time it is migrated to the nearest bank to the processor, thereby preventing movement of data between distant banks.

Nevertheless, there is another problem to be resolved during the data migration, which concerns on to what should be done with the victim blocks from the nearest bank which will be replaced with a new block. To solve this, there are two main policies which address the problem. The first policy removes the victim block from the cache. On the second policy the victim block changes place with the block to take its place. Thus, it is avoided that the block which could be used again, to be removed from the cache, making it move more slowly out of the cache according to the time that it is not being used.

7. Conclusion

Multi-core processors are a reality nowadays and new solutions for performance improving are still necessary on future architectures techniques as for the internal communication network and the whole system involving the cache memory.

This paradigm change in processor architecture is just possible with the constant increase in the integration technology and continuous miniaturization of manufacturing processes, thereby generating an increase in available resources within the chip. Focus on physical problems of new technologies, the non-uniform cache architectures are strong candidates to be used in systems dominated by access latencies, wire delays, etc. Opening new horizons for researchs which improves the performance of applications providing large computational resources.

This paper presented a survey on non-uniform cache architectures, regarding its architecture, the data fetch, data mapping and others.

Observing the researches developed on NUCA memories, we can indicate some examples of possible studies, such as: Evaluation of reconfigurable interconnection networks and study the adaptation to the context of NUCA memories; Evaluation of methods to reduce the delay, improving the migration policies on D-NUCA models; Study mechanisms to support the control of NUCA memory consistency in multi-cores; Study of contention impact on proposed solutions looking for bottlenecks created by main memory access time by NUCA memories. Use of techniques from transactional memories aggregated to NUCA in order to ensure the a consistent system.

References

[1] N. Aggarwal et al. Isolation in commodity multicore processors. *Computer*, 40(6):49–59, 2007.

[2] M. A. Z. Alves, H. C. Freitas, and P. O. A. Navaux. Investigation of shared l2 cache on many-core processors. In *Proceedings Workshop on Many-Core*, pages 21–30, Berlin, 2009. VDE Verlag GMBH.

[3] A. Bardine, M. Comparetti, P. Foglia, G. Gabrielli, C. A. Prete, and P. Stenströ andm. Leveraging data promotion for low power d-nuca caches. In *Digital System Design Architectures, Methods and Tools, 2008. DSD '08. 11th EUROMICRO Conference on*, pages 307–316, Sept. 2008.

[4] C. Kim, D. Burger, and S. Q. Keckler. An adaptive, non-uniform cache structure for wire-delay dominated on-chip-caches. In *Proceedings Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, pages 211–222. IEEE, 2002.

[5] Shyamkumar Thoziyoor and Naveen Muralimanohar and Norman P. Jouppi. Cacti 5.0. *HP Laboratories*, 2007. Disponível em: <http://www.hpl.hp.com/techreports/2007/HPL-2007-167.html>. Acesso em: junho 2009.