Software Prepromotion for Non-Uniform Cache Architecture

Junjie Wu
National laboratory for parallel and distributed processing, Changsha, China
Email: junjie.ben@gmail.com

Xiaohui Pan
National laboratory for parallel and distributed processing, Changsha, China
Email: xiaohuipan@hotmail.com

Xuejun Yang
National laboratory for parallel and distributed processing, Changsha, China
Email: xjyang@nudt.edu.cn

***Abstract*—As a solution to growing global wire delay, non-uniform cache architecture (NUCA) has already been a trend in large cache designs. The access time of NUCA is determined by the distance between the cache bank containing the required data and the processor. Thus, one of the important NUCA researches focuses on how to place data to be used into cache banks close to the processor. This paper proposes software prepromotion technique, which prepromote data using prepromotion instructions as similar as software prefetching does. Besides the basic software prepromotion, this paper also proposes *smart multihop software prepromotion* (SMSP), *very long software prepromotion* (VLSP) and their combination technique. SMSP intelligently chooses cache banks which the prepromoted data most ideally suit to being moved into. And VLSP prepromote multiple data using one instruction. Finally, we evaluate our approaches by testing 7 kernel benchmarks on a full-system simulator. The basic software prepromotion gets an average improvement of 2.6893% in IPC. The SMSP improves IPC by 7.0928% averagely. And the VLSP gets an IPC improvement of 7.2194% averagely. Lastly, after combining the SMSP and VLSP, the average improvement in IPC achieves 11.8650%.**

***Index Terms*—NUCA, software prepromotion, smart multihop software prepromotion, very long software prepromotion, prefetching**

I. INTRODUCTION

The speed gap between processors and memories has always been the performance bottleneck of computer systems. Cache is one of the key technologies for alleviating the memory wall problem. Thus, researches on caches have always been very concerned. Along with the development of microelectronics technology, more and more transistors can be integrated on one chip. Researchers begin to study caches with large capacity on chips. When the capacity of caches becomes larger and larger, the delay of global wires on chips increases higher and higher. If we still use the traditional cache architecture, the access time of the whole cache has to accommodate itself to the delay of the farthest cache bank. Hence, a kind of Non-Uniform Cache Architecture (NUCA) is proposed to bridge the large cache capacity and the low access latency [1]. The access time of NUCA may be one of many different values according the distance between the processor and the accessed cache bank. When the processor accesses a near cache bank, the access latency is low. And when the processor accesses a far one, the time is long. The arising of NUCA brings large cache designs new energy.

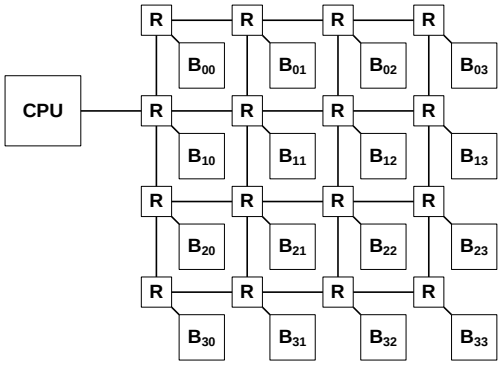


Figure 1. An example of NUCA

There are many cache banks physically distributed in NUCA. And these banks are connected through an on-chip interconnection network. When a memory access instruction is executed, the access package routes in the interconnection network. Thus, if the data accessed stay in a bank far from the processor, the package will work through many lines and routers. Thus, one of the key problems is how to make the accessed data closer to the processor. Kim et al. proposed the promotion technique, which can moves data to the banks near processors [1]. When a data is hit, the promotion mechanism is triggered

doi:10.4304/jsw.5.1.11-19

and the data accessed is moved one step to the processor. When the same data is accessed next time, the access latency is reduced. However, the promoted data may not be accessed in the future. This problem results from the lagging of promotion behind accessing.

```
for (i=0; i<N; i++) {
   sum += a[i];
}
```

Figure 2. A loop segment

To solve the lagging problem, this paper proposes software prepromotion for non-uniform cache architecture. The prepromoted target is the data which may be accessed in the near future, while the promoted one is the data which was accessed recently. Prepromotion is similar to prefetching. A prefetching operation moves a data to a memory hierarchy closer to the processor, while a prepromotion one moves it to a cache bank near the processor. Different from hardware ones in [2, 3], software prepromotion is a technique in which programmers or compilers add prepromotion instructions in the program explicitly. Since programmers and compilers can get more program information through static analyzing than hardware, the software prepromotion often acts more precisely than the hardware one.

The main contributions of this paper include:

1. Software prepromotion for non-uniform cache architecture is proposed. To our knowledge, this is the first work that proposes and studies the software prepromotion.
2. We propose smart multihop software prepromotion, which can improve the prepromotion performance according to the characteristics of NUCA.
3. Very long software prepromotion is proposed for reducing prepromotion instructions and saving bandwidth utilization of interconnection networks.

The rest of this paper is organized as follows. Section 2 proposes basic software prepromotion. Section 3 introduces multihop software prepromotion, and then proposes smart multihop software prepromotion. Section 4 studies very long software prepromotion. Section 5 and section 6 evaluate all the software prepromotion proposed in this paper. Finally, section 7 illustrates related work and we conclude in section 8.

II. Basic Software Prepromotion

Fig. 1 shows an example of NUCA. In this example, cache is consists of 16 banks, which corresponds with 16-way set-associativity. Each way in a set locates one cache bank. All banks are connected by an interconnection network. When the processor issues a memory access instruction, the memory request package spreads in the network. If some bank has the data requested, it will send

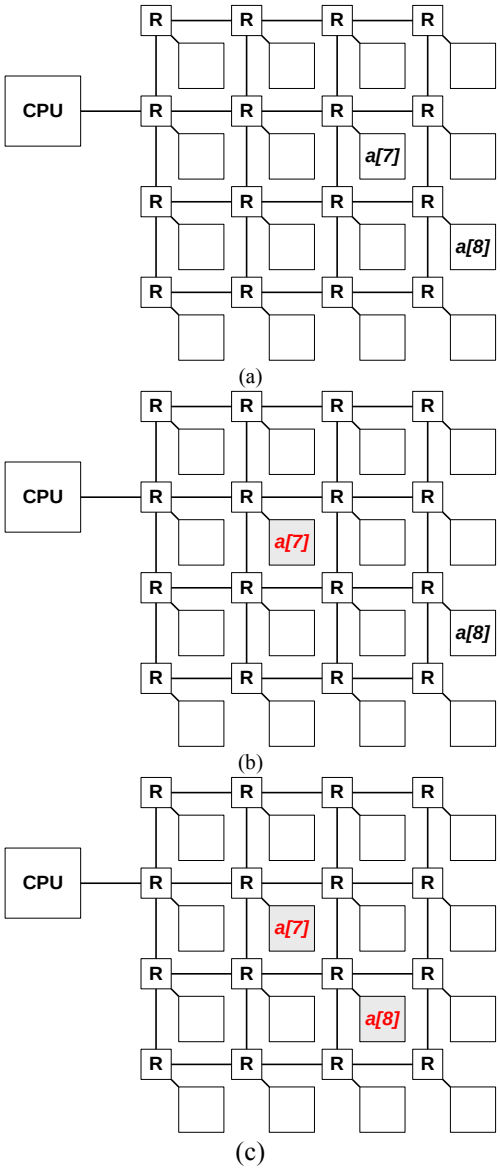


Figure 3. Promotion and prepromotion
(a) Before promotion and prepromotion
(b) When the i=7 iteration is over with promotion
(c) When the i=7 iteration is over with prepromotion

back the acknowledge package after receiving the requested one. Thus, the distance between the processor and the cache bank that contains requested data determines the access time of NUCA.

To distinguish between prepromotion and promotion, we study the loop in Fig. 2. In this loop, all elements of array a will be accessed sequentially. We assume that the array a has been loaded in the cache before this loop. When the loop runs at the iteration with i=7, the processor issues the memory reference instruction accessing a[7], which is staying in the bank shown in Fig. 3a. If the promotion mechanism is adopted, the element a[7] will be moved into another bank as shown in Fig. 3b.

This work is supported in part by the National Natural Science Foundation of China under Grant No. 60621003 and No. 60873014, and the National High-Tech Research and Development Plan of China ("863" plan) under Grant No. 2007AA12Z147. The corresponding author: Junjie Wu. Email: junjie.ben@gmail.com

If we adopt prepromotion mechanism, the element a[8] may be promote as well in this iteration running. In conclusion, the target element of promotion is the hit data, while the target one of prepromotion is the data to be accessed in the near future.

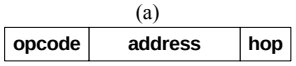
```
for (i=0; i<N; i++) {
   prepromote(&a[i+1]);
   sum += a[i];
}

for (i=0; i<N; i+=4) {
   prepromote(&a[i+4]);
   sum += a[i];
   sum += a[i+1];
   sum += a[i+2];
   sum += a[i+3];
}

prepromote(&a[0]);

for (i=0; i<N-4; i+=4) {
   prepromote(&a[i+4]);
   sum += a[i];
   sum += a[i+1];
   sum += a[i+2];
   sum += a[i+3];
}

for (; i<N; i++) {
   sum += a[i];
}
```

Figure 4. Software prepromotion examples

Prepromotion is similar to prefetching, and both them are in non-blocking modes. That is to say, a prepromotion instruction has not to wait the completion of the prepromotion operation. This kind of non-blocking way allows the parallelism between computing and memory accessing, so it can hide the delay of memory accessing effectively. Since a cache is constituted by cache lines, the granularity of prepromotion is cache line. The unrolled loop in Fig. 4b shows the prepromotion code taking cache line into account. However, the element a[0] will not be prepromoted and the prepromotion in the last iteration will be wasteful in Fig. 4b. To solve these problems, we can use software pipeline to generate the code as shown in Fig. 4c, where all elements of the array a will prepromoted.

Prepromote(addr, hop)

(a)

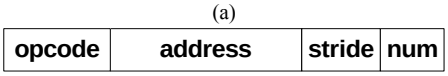


(b)

Figure 5. Multihop software prepromotion instruction
(a) Assembly code (b) Instruction format

Prepromote(addr, stride, num)

(a)



(b)

Figure 7. VLSP instruction
(a) Assembly code
(b) Instruction format

Although prepromotion is similar to prefetching, it faces its own problems because of its different platform from prefetching. One problem is about the prepromotion destination. There are many banks in a non-uniform cache, and anyone of them can be the destination of a prepromoted data. Thus, we studies multihop software prepromotion in section 3. The other problem is how to use the interconnection network of NUCA in a high efficient way. In section 4, we propose very long prepromotion, which not only saves the bandwidth of the interconnection network but also reduces the prepromotion instructions in programs.

III. SMART MULTIHOP SOFTWARE PREPROMOTION

In a NUCA, a prepromoted data can be moved into one of multiple banks, while the basic software prepromotion only prepromote the data in one step. In this section, we propose multihop software prepromotion. Multihop software prepromotion can prepromote a data into the bank far from the source bank in a given distance. Hence, we add the hop field in the prepromotion instruction as illustrated in Fig. 5.

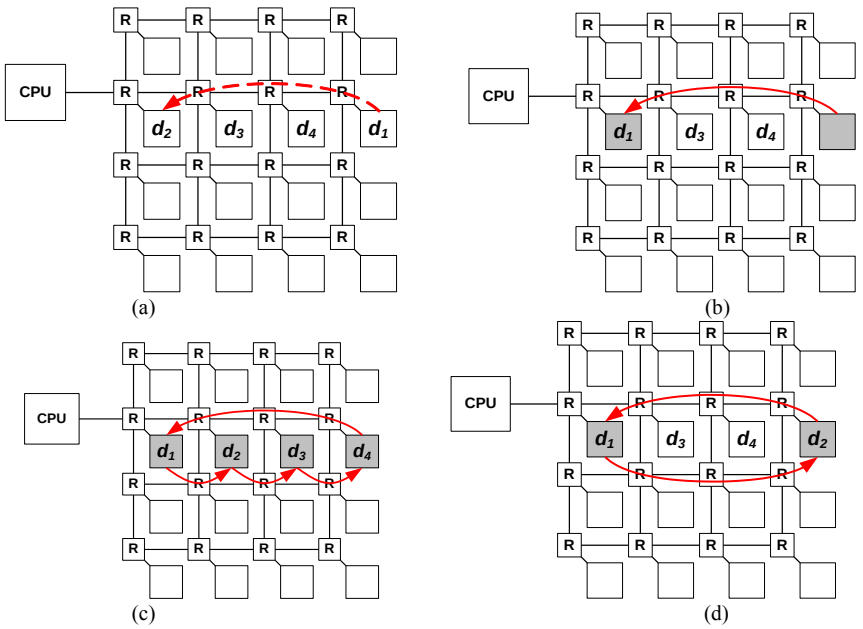


Figure 6. Three different strategies for multihop prepromotion
(a) The bank containing d1 is to be prepromote to the bank containing d2
(b) After prepromotion with the replacement strategy
(c) After prepromotion with the degradation strategy
(d) After prepromotion with the exchanging strategy

There are two aspects to be considered. One is how to handle the useful data in the destination bank. In Fig. 6a, the data d1 will be prepromoted into the bank containing d2. The simplest way is to evict the data d2 out of the cache as shown in Fig. 6b. We call this method *replacement strategy*. If the evicted data is still to be used in the near future, the replacement strategy will lead to cache miss. The second strategy is to degrade the data in one step. In this way, all the valid data between the source one and the destination one should be

degraded in one step as shown in Fig. 6c. We name it *degradation strategy*. The degradation strategy often brings up too much data moving, which will consume much bandwidth of the interconnection network. Another compromising method is *exchanging strategy*. The exchanging strategy only exchanges the data in source bank and destination one, and does not move data in other banks. Thus, the exchanging strategy not only prevents data from being evicted out of cache, but also saves the network bandwidth.

```
prepromote(&a[0], 1, 2);

for (i=0; i<N-8; i+=8) {
   prepromote(&a[i+8], 1, 2);
   sum += a[i];
   sum += a[i+1];
   sum += a[i+2];
   sum += a[i+3];
   sum += a[i+4];
   sum += a[i+5];
   sum += a[i+6];
   sum += a[i+7];
}

for (; i<N; i++) {
   sum += a[i];
}
```

Figure 8. An example of VLSP

package head	address	stride	num	PP bit vector

Figure 9. Interconnection network package for VLSP

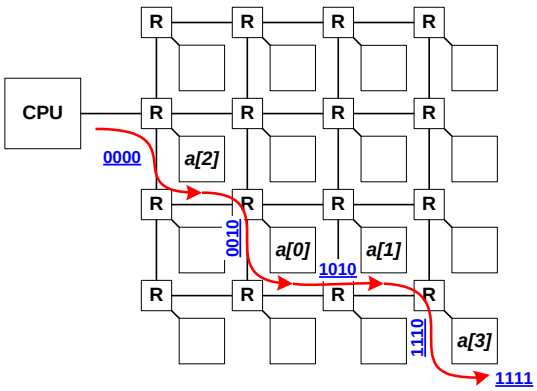


Figure 10. An VLSP package transmits in the interconnection network (PP bit vector in this VLSP is underlined in the figure)

The other problem is how to choose the prepromotion hop. Generally, the nearer bank to the processor is the most frequent accessed data moved into; the lower is the access time. However, because the bank closer to the processor often contains more frequent accessed data, the large hop prepromotion may make the useful data far from the processor. This is the so-called pollution problem. On the other hand, there may be no banks far from the source bank in the given distance. And even if the destination bank exists, there may be two or more banks with the same distance. The choosing of the destination bank will directly affect the prepromotion performance. To deal with this problem, we propose *smart multihop software prepromotion* (SMSP). When a SMSP package walks in the interconnection network, it records the information of banks in its path. The banks in its path are just those between the source bank and the processor. Thus, the destination bank will be one of them. We choose the destination in these candidate banks according to different priority level. The first is the banks containing no valid data. And if there are two or more banks satisfying this condition, we choose the closest bank to the processor as the destination of prepromotion. The second is the banks containing clean data. A clean data is often read by the processor. The higher memory hierarchies may contain the copy of this clean data. So, clean data is accessed with lower probability than dirty data. If there are two or more clean ones in the path, we choose the bank farthest from the processor. If there is neither empty room nor clean data, the SMSP will prepromote the data in only one step.

IV. Very Long Software Prepromotion

Similar with software prefetching, software prepromotion has to add many prepromotion instructions in programs. If these instructions are too excessive, not only will the execution time of whole program be affected, but also will too much cost come arise. The cost includes bandwidth increasing, package conflict, power consumption raising, etc. To reduce the cost brought by too much software instructions, we propose *very long software prepromotion* (VLSP).

The basic idea of VLSP is completing prepromotion of multiple data with one prepromotion instruction. On one hand, the coming of routers in NUCA's interconnection networks brings more functions than the traditional cache. On the other hand, cache banks are distributed in NUCA, so many banks can be accessed simultaneously. VLSP makes full use of these features of NUCA to prepromote multiple data with one instruction.

We extend the format of the basic software prepromotion instruction to implement VLSP. As illustrated in Fig. 7. Stride field and num field are added in VLSP instruction. An instruction Prepromote(addr, stride, num) simultaneously promote data which addresses are addr, addr+stride, addr+2stride, …, addr+(num-1)stride.

Fig. 8 shows the code when we use VLSP in the program shown in Fig. 2 and Fig. 4. Here, we use one

Table I. Experimental Configuration

Processor	UltraSPARC-III, in-order
L1 I-Cache	32KB, 64B block size, 4-way set-associative
L1 D-Cache	32KB, 64B block size, 4-way set-associative
L2 (NUCA)	8MB, 64B block size, 16-way set-associative
OS	Solaris 10

prepromotion instruction to prepromote two cache lines, which can contain 8 elements of the array a. Thus, the loop is unrolled 8 times to match the VLSP operations. It is obvious that the number of prepromotion instructions in VLSP is reduced to the half.

TABLE II. L2 NUCA CACHE CONFIGURATION DERIVED BY CACTI 6.0

Layout	4 rows × 4 columns
Horizontal Wire Delay	2 cycles
Vertical Wire Delay	2 cycles
Router Delay	3 cycles
Bank Access Delay	6 cycles

If we want to prepromote n data items using the basic prepromotion instruction, we need issue n instructions. To complete the prepromotion, each instruction will produce request package. However, if we use the VLSP instruction, only one instruction is needed and only one package will be produced in the interconnection network. The package format is shown in Fig. 9. The PP bit vector in Fig. 9 denotes which data has been prepromoted. When the prepromotion package passes some bank, the router beside this bank is in charge of computing the data addresses to be prepromoted and issuing the corresponding operations to this bank. If a bank contains the data requested for prepromotion, the router sets the corresponding bit of the PP bit vector in the package to 1. Once all the bits of the PP bit vector are set to 1, the package will not be transmitted again. Fig. 10 shows an example in which a VLSP package transmits in the interconnection network. It is obvious that the VLSP can reduce the number of prepromotion packages efficiently under the help of routers. Meanwhile, because all banks are distributed, those prepromotion operations can be completed simultaneously.

In actual, SMSP and VLSP can be combined. We test the combination in the experiment. As seen in section 6, the combination of SMSP and VLSP achieves the best performance.

V. EVALUATION METHODOLOGY

To evaluate the software prepromotion proposed in this paper, we test seven kernel benchmarks in a full-system simulator, Simics [4]. The Table I lists the experimental configuration. The target of prepromotion is a L2 NUCA cache. It is 16-way set-associative. And we derive its NUCA parameter using CACTI 6.0 [5] as illustrated in Table II. There are 16 cache banks in it, and they are organized as 4 rows and 4 columns. Each way in a set locates in its own cache bank. The prepromotion operations act between two cache banks, which belong to the same set. Thus, prepromotion does not affect the correctness of the program running.

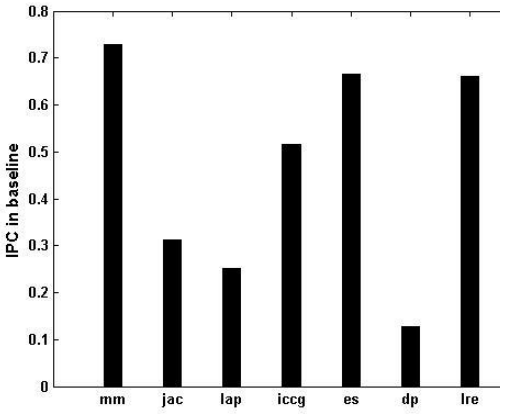


Figure 11. IPC in the baseline

TABLE III. BENCHMARKS CHARACTERISTICS

Benchmark	Description	# of Instructions	# of L2 Access
mm	Matrix multiplication	30.50M	0.44M
jac	Jacobi iteration algorithm	32.79M	1.71M
lap	Laplace transformation	53.15M	3.79M
iccg	Incomplete Cholesky Conjugate Gradient	24.70M	0.71M
es	Equation of state fragment	47.80M	0.74M
dp	Difference predictors	48.78M	8.86M
lre	General linear recurrence equations	44.62M	0.62M

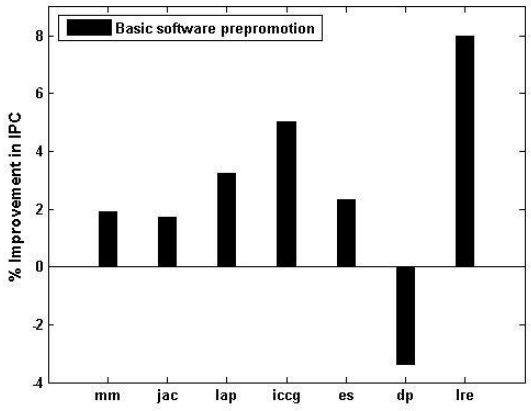


Figure 12. IPC improvement of basic software prepromotion over baseline

Table III shows the benchmarks used in our experiment. Iccg, es, dp and lre are all loops from the Livermore benchmark [6]. Table III lists the numbers of instructions and L2 cache accesses. All benchmarks are compiled by GCC 4.2 with -O3 optimizing option.

We test the performance of our approaches in detail, and choose the system without software prepromotion as

our baseline. Fig. 11 shows the IPC of benchmarks in the baseline. The average of IPC is 0.4661.

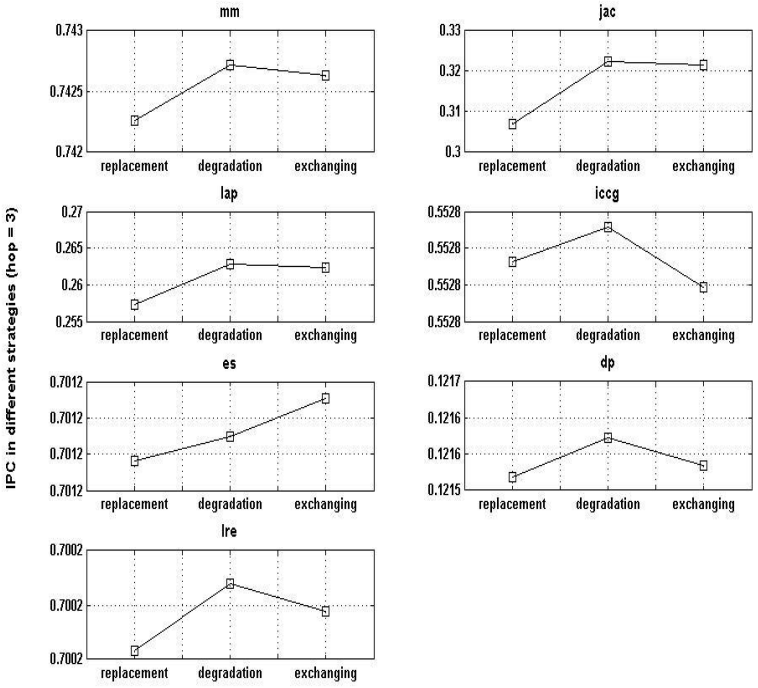


Figure 13. IPC in multihop prepromotion with different strategies

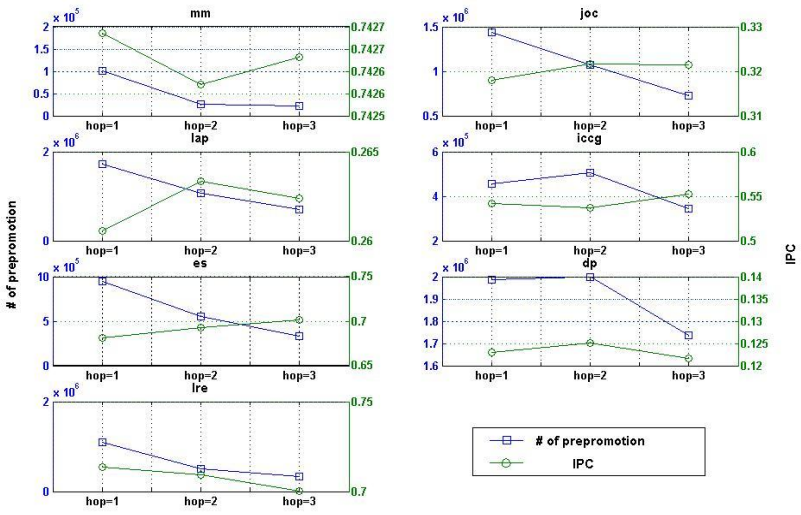


Figure 14. Number of prepromotion and IPC in multihop prepromotion with different hops

VI. RESULTS AND ANALYSIS

A. Basic software prepromotion

Fig. 12 shows the improvement using basic software prepromotion in IPC over the baseline. It is obvious that most benchmarks except dp gets the performance improvement. The average improvement in IPC achieves 2.6893%. Dp is the difference predictors kernel. One of its loops accesses two arrays in column major mode. Since each row of those loops contains 1024 elements, multiple data accessed continuously are mapped into one set of the cache. Thus, the performance of dp is the lowest as shown in Fig. 11. The basic software prepromotion sharpens the problems because the prepromoted data compete more seriously. The problem reflects some intrinsic shortcomings in the basic software prepromotion. The blind prepromotion may replace some useful data far from the processor. On the other hand, the adding of prepromotion instructions may bring more cost than the gain.

B. Multihop software prepromotion

Compared with basic software prepromotion, multihop prepromotion can get better performance. Fig. 13 shows the IPC of benchmarks with hop = 3 in different strategies. The degradation management gets the best performance in most benchmarks except es. The average IPC in the replacement management is 0.4832; the one in the degradation management is 0.4862; and that in the exchanging management is 0.4860. The performance of the exchanging strategy is very close to that of the degradation one. However, the exchanging strategy saves lots of bandwidth compared degradation. Thus, the exchanging strategy is our first choice. In the following experiments, we choose exchanging as the management strategy of multihop software prepromotion.

To study the performance influence from different hops, we compare the number of prepromotion operations and system IPC in different hops as shown in Fig. 14. Note that the IPC does not increase along with the increment of the hop value. This appearance results from the decrement of prepromotion operations. A data to be prepromoted with hop = n may not be prepromoted because there may be no bank that is away from the source bank at a distance of n. On the other hand, we find that different benchmarks favor different hop values in Fig. 14. Mm and lre favor the prepromotion with hop = 1; joc, lap and dp gets the best performance at hop = 2; while the performance of

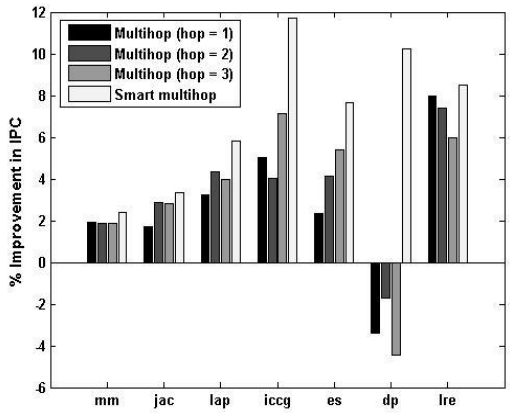


Figure 15. IPC improvement of basic multihop and smart multihop software prepromotion

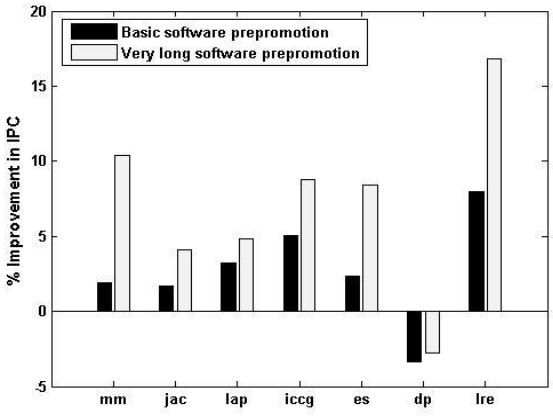


Figure 16. IPC improvement of very long software prepromotion

prepromotion with hop = 3 outperforms the other two in iccg and es. Thus, we need an intelligent method which can select the best hop value for different benchmarks.

The smart multihop software prepromotion is just the best candidate for solve the above problem. Fig. 15 illustrates the IPC improvement of the basic multihop and smart multihop over the baseline. It is obvious that the smart multihop prepromotion achieves the best performance. The average improvement of IPC in the smart one is 7.0928%, while those in the basic multihop are 2.6893%, 3.2756% and 3.2534% respectively from hop = 1 to hop = 3. It is exciting that the smart prepromotion gets a considerable performance improvement in dp, which is hard to optimized using basic software prepromotion and basic multihop software prepromotion.

C. Very long software prepromotion

Fig. 16 shows the improvement of IPC with basic software prepromotion and VLSP over the baseline. In the experiments, we use one VLSP instruction to prepromote four cache lines. By reducing instructions in programs and packages generated in networks, the very long software prepromotion gets better performance than the basic one. The average of IPC improvement with VLSP arrives at 7.2194%, while that with the basic prepromotion is 2.6893%.

As shown in Fig. 16, the VLSP can not prevent the blind prepromotion as well, so dp with the VLSP does not get the performance improvement. Thus, we integrate the smart multihop software prepromotion and the very long software prepromotion. As illustrated in Fig. 17, the combination gets the better performance. The average improvement in IPC with the combination achieves 11.8650%. And dp also obtain the considerable performance improvement.

VII. Related Work

The software prepromotion proposed in this paper prepromote data in the non-uniform cache architecture. We summarize the work that most closely relates to the techniques proposed in this paper.

A. Non-uniform cache architecture

Changkyu Kim firstly proposed non-uniform cache architecture in 2002 [1]. After that, lots of researches emerged [7-14]. There are two kinds of NUCA, static NUCA and dynamic NUCA. The static NUCA fixes each data in one cache bank, while the dynamic one permits that data stay in one of multiple banks. Thus, data can be moved between banks in the dynamic NUCA. This is the so-called promotion. It is used to reduce the NUCA access time. However, promotion acts when the data is accessed next time. Thus, promotion technique belongs to a passive method. The prepromotion is an active technique, which prepromote data that may be used in the near future. Akio Kodama proposed one block ahead (OBL) prepromotion [2], which belongs to hardware prepromotion. To our knowledge, the software prepromotion proposed in this paper is the first work that studies prepromotion using software methods. Under the help of programmer or compiler, the software prepromotion is often saner than the hardware one, so it can get better performance. Besides, the smart software prepromotion and the very long software prepromotion proposed in this paper overcome the shortcoming of the basic software method.

B. Software prefetching

Prefetching is one of the most important techniques for hiding the memory access latency. Prefetching is divided into three classes: hardware prefetching [15], software prefetching [16-18] and the combination [19]. Hardware prefetching uses hardware to predict the future accesses using the history information of programs. Software prefetching inserts prefetching instructions in programs. The software prepromotion proposed in this paper is similar to software prefetching. Thus, some optimizing approaches for software prefetching are also adapted to our software prepromotion. However, because of the different platforms of prefetching and prepromotion, software prepromotion is different from software prefetching. For example, the smart multihop prepromotion and the very long software prepromotion proposed in this paper just act for software prepromotion, and there is no similar technique in software prefetching.

C. Interconnection network

Interconnection network is one of the key techniques in non-uniform cache architecture. This kind of network is an on-chip one. The approaches in this paper need some support of the interconnection network. However, compared with the on-chip network used in multi-core processor [20-24], the modified part is very little.

VIII. Conclusion and Future Work

Non-uniform cache architecture has always been the trend of the future cache design. This paper has proposed the concept of software prepromotion and studied software prepromotion in detail. Besides the basic software prepromotion, smart multihop one has been proposed to match the particular of NUCA. And very long software prepromotion has also been studied for reducing the number of prepromotion instructions and improving the utilization of network bandwidth. Finally, we have tested seven kernel benchmarks to evaluate our approaches. The smart multihop software prepromotion improves IPC by 7.0928% averagely. And the very long software prepromotion gets the IPC improvement of 7.2194% averagely. After combining smart multihop and very long software prepromotion, the improvement in IPC achieves 11.8650% averagely. Similar with software prefetching, software prepromotion lets programmers or compilers to insert prepromotion instructions in programs. How to do the work using compilers automatically is the challenge. This and the optimizing methods for software prepromotion are our future work.

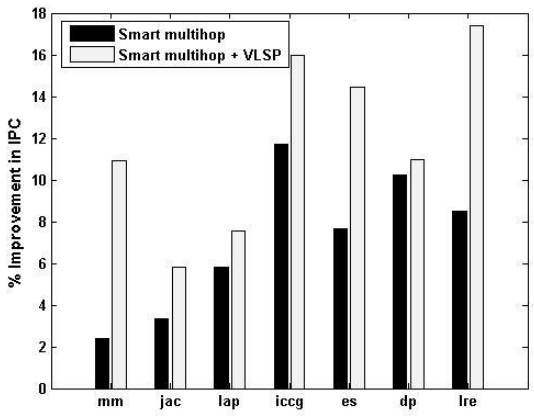


Figure 17. IPC improvement of the combination between smart multihop prepromotion and VLSP

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant No. 60621003 and No. 60873014, and the National High-Tech Research and Development Plan of China ("863" plan) under Grant No. 2007AA12Z147.

Reference

[1] C. Kim, D. Burger, and S. W. Keckler, “An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches,” in ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems. New York, NY, USA: ACM, 2002, pp. 211–222.

[2] Kodama, T. Sato. “A non-uniform cache architecture on networks-on-chip: A fully associative approach with pre-promotion,” in ISIC ’04:10th International Symposium on Integrated Circuits, Devices and Systems, 2004, CD-ROM.

[3] J. Wu, X. Yang. “Pre-promotion with Arbitrary Strides in Non-Uniform Caches,” Electronic Journal of China, unpublished.

[4] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. H. llberg, J. H¨ogberg, F. Larsson, A. Moestedt, and B. Werner, “Simics: A full system simulation platform,” Computer, vol. 35, no. 2, pp. 50–58, 2002.

[5] N.Muralimanohar, R. Balasubramonian, and N. Jouppi, “Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0,” in MICRO ’07: Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture. Washington, DC, USA: IEEE Computer Society, 2007, pp. 3–14.

[6] F. H. McMahon. “Livermore fortran kernels: A computer test of numerical performance range,” Technical Report UCRL-53745, Lawrence Livermore National Laboratory, Livermore, CA, December 1986.

[7] Z. Chishti, M. D. Powell, and T. N. Vijaykumar, “Distance associativity for high-performance energy-efficient non-uniform cache architectures,” in MICRO 36: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture. Washington, DC, USA: IEEE Computer Society, 2003, p. 55.

[8] B. M. Beckmann and D. A. Wood, “Managing wire delay in large chip-multiprocessor caches,” in MICRO 37: Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture. Washington, DC, USA: IEEE Computer Society, 2004, pp. 319–330.

[9] Z. Chishti, M. D. Powell, and T. N. Vijaykumar, “Optimizing replication, communication, and capacity allocation in cmps,” in ISCA ’05: Proceedings of the 32nd annual international symposium on Computer Architecture. Washington, DC, USA: IEEE Computer Society, 2005, pp. 357–368.

[10] Bardine, P. Foglia, G. Gabrielli, C. A. Prete, and P. Stenstr¨om, “Improving power efficiency of d-nuca caches,” SIGARCH Comput. Archit. News, vol. 35, no. 4, pp. 53–58, 2007.

[11] Bardine, P. Foglia, G. Gabrielli, and C. A. Prete, “Analysis of static and dynamic energy consumption in nuca caches: initial results,” in MEDEA ’07: Proceedings of the 2007 workshop on MEmory performance. New York, NY, USA: ACM, 2007, pp. 105–112.

[12] Bardine, M. Comparetti, P. Foglia, G. Gabrielli, C. A. Prete, and P. Stenstr¨om, “Leveraging data promotion for low power d-nuca caches,” in DSD ’08: Proceedings of the 2008 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools. Washington, DC, USA: IEEE Computer Society, 2008, pp. 307–316.

[13] J. Merino, V. Puente, P. Prieto, and J. ´ Angel Gregorio, “Sp-nuca: a cost effective dynamic non-uniform cache architecture,” SIGARCH Comput. Archit. News, vol. 36, no. 2, pp. 64–71, 2008.

[14] M. Kandemir, F. Li, M. J. Irwin, and S. W. Son, “A novel migration-based nuca design for chip multiprocessors,” in SC ’08: Proceedings of the 2008 ACM/IEEE conference

on Supercomputing. Piscataway, NJ, USA: IEEE Press, 2008, pp. 1–12.
[15] J.-L. Baer and T.-F. Chen, "Effective hardware-based data prefetching for high-performance processors," IEEE Trans. Comput., vol. 44, no. 5, pp. 609–623, 1995.
[16] D. Bernstein, D. Cohen, and A. Freund, "Compiler techniques for data prefetching on the powerpc," in PACT '95: Proceedings of the IFIP WG10.3 working conference on Parallel architectures and compilation techniques. Manchester, UK, UK: IFIP Working Group on Algol, 1995, pp. 19–26.
[17] V. Santhanam, E. H. Gornish, and W.-C. Hsu, "Data prefetching on the hp pa-8000," SIGARCH Comput. Archit. News, vol. 25, no. 2, pp. 264–273, 1997.
[18] K. C. Yeager, "The mips r10000 superscalar microprocessor," IEEE Micro, vol. 16, no. 2, pp. 28–40, 1996.
[19] E. H. Gornish and A. Veidenbaum, "An integrated hardware/software data prefetching scheme for shared-memory multiprocessors," Int. J. Parallel Program., vol. 27, no. 1, pp. 35–70, 1999.
[20] N. Eisley, L.-S. Peh, and L. Shang, "In-network cache coherence," in MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture. Washington, DC, USA: IEEE Computer Society, 2006, pp. 321–332.
[21] D. Park, R. Das, C. Nicopoulos, J. Kim, N. Vijaykrishnan, R. Iyer, and C. R. Das, "Design of a dynamic priority-based fast path architecture for on-chip interconnects," in HOTI '07: Proceedings of the 15th Annual IEEE Symposium on High-Performance Interconnects. Washington, DC, USA: IEEE Computer Society, 2007, pp. 15–20.
[22] W. J. Dally, "Interconnect-centric computing," in HPCA '07: Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture. Washington, DC, USA: IEEE Computer Society, 2007, p. 1.
[23] N. Aggarwal, P. Ranganathan, N. P. Jouppi, and J. E. Smith, "Configurable isolation: building high availability systems with commodity multi-core processors," in ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture. New York, NY, USA: ACM, 2007, pp. 470–481.
[24] D. Tutsch and M. Malek, "Comparison of network-on-chip topologies for multicore systems considering multicast and local traffic," in Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 1–9.

Junjie Wu, Anhui, China, 1981. He is a Ph.D. degree candidate in computer science and technology from National University of Defense Technology, Changsha Hunan, China. And he received his master degree in computer science and technology from the same university.

He studied processor architecture, compilation technique, quantum computer architecture and volume holographic storage. And his current research interests focus on cache studies. He has published many papers in the top Chinese and international journals and conferences in computer science. Now, he is studying at the National Laboratory for Parallel and Distributed processing in National University of Defense Technology for his Ph.D. degree.

Mr. Wu is a student member of IEEE, ACM and China Computer Federation. He has been invited to publish a paper in the Journal of Computer Research and Development when it has started publication for 50 years.

Xiaohui Pan, Shanxi, China, 1977. She is an assistant professor of National University of Defense Technology, Changsha Hunan, China. And she received her master degree in computer science and technology from the same university.

She studied virtual reality, computer graphics and computer emulation. And her current research interests focus on computer architecture and computer application. She has taught for years, and now she is still teaching in National University of Defense Technology.

Ms. Pan is a member of China Computer Federation. She has published many papers in computer science and computer educations.

Xuejun Yang, Shandong, China, 1963. He is a professor and doctoral supervisor of National University of Defense Technology, Changsha, Hunan, China. He received his Ph.D. degree in computer science and technology from National University of Defense Technology.

He studied parallel computer architecture, parallel compilation technology, parallel operating system, etc. And his current research interests focus on cache studies, stream processing, fault tolerant, and nano-computing. He has published over one hundred papers in the top international journals and conferences including IEEE Trans. on Parallel and Distributed Systems, Journal of Supercomputing, ISCA, PACT, PPoPP, ICDCS, etc.

Prof. Yang is a senior member of IEEE, ACM and China Computer Federation. He is the governor of China Computer Federation. He is the top young experts of computer profession in China. He is granted by the Innovative Research Group Foundation of the National Natural Science Foundation of China. He has been invited to give a keynote speech in China National Computer Conference (CNCC'08). And he is also invited to give a keynote speech in ISPA'09.