Characteristics of an On-Chip Cache on NEC SX Vector Architecture

Akihiro MUSA¹,³*, Yoshiei SATO¹,³, Ryusuke EGAWA¹,², Hiroyuki TAKIZAWA¹,², Koki OKABE³, and Hiroaki KOBAYASHI¹,²

¹Graduate School of Information Sciences, Tohoku University, Sendai 980-8578, Japan
²Cyberscience Center, Tohoku University, Sendai 980-8578, Japan
³NEC Corporation, Tokyo 108-8001, Japan

Received May 6, 2008; final version accepted December 10, 2008

Thanks to the highly effective memory bandwidth of the vector systems, they can achieve the high computation efficiency for computation-intensive scientific applications. However, they have been encountering the memory wall problem and the effective memory bandwidth rate has decreased, resulting in the decrease in the bytes per flop rates of recent vector systems from 4 (SX-7 and SX-8) to 2 (SX-8R) and 2.5 (SX-9). The situation is getting worse as many functions units and/or cores will be brought into a single chip, because the pin bandwidth is limited and does not scale. To solve the problem, we propose an on-chip cache, called vector cache, to maintain the effective memory bandwidth rate of future vector supercomputers. The vector cache employs a bypass mechanism between the main memory and register files under software controls. We evaluate the performance of the vector cache on the NEC SX vector processor architecture with bytes per flop rates of 2 B/FLOP and 1 B/FLOP, to clarify the basic characteristics of the vector cache. For the evaluation, we use the NEC SX-7 simulator extended with the vector cache mechanism. Benchmark programs for performance evaluation are two DAXPY-like loops and five leading scientific applications. The results indicate that the vector cache boosts the computational efficiencies of the 2 B/FLOP and 1 B/FLOP systems up to the level of the 4 B/FLOP system. Especially, in the case where cache hit rates exceed 50%, the 2 B/FLOP system can achieve a performance comparable to the 4 B/FLOP system. The vector cache with the bypass mechanism can provide the data both from the main memory and the cache simultaneously. In addition, from the viewpoints of designing the cache, we investigate the impact of cache associativity on the cache hit rate, and the relationship between cache latency and the performance. The results also suggest that the associativity hardly affects the cache hit rate, and the effects of the cache latency depend on the vector loop length of applications. The cache shorter latency contributes to the performance improvement of the applications with shorter loop lengths, even in the case of the 4 B/FLOP system. In the case of longer loop lengths of 256 or more, the latency can effectively be hidden, and the performance is not sensitive to the cache latency. Finally, we discuss the effects of selective caching using the bypass mechanism and loop unrolling on the vector cache performance for the scientific applications. The selective caching is effective for efficient use of the limited cache capacity. The loop unrolling is also effective for the improvement of performance, resulting in a synergistic effect with caching. However, there are exceptional cases; the loop unrolling worsens the cache hit rate due to an increase in the working space to process the unrolled loops over the cache. In this case, an increase in the cache miss rate cancels the gain obtained by unrolling.

KEYWORDS: Vector processing, Vector cache, Performance characterization, Memory system, Scientific application

1. Introduction

Vector supercomputers have high computation efficiency for computation-intensive scientific applications. In our previous work (Kobayashi (2006), Musa et al. (2006)), we have shown that NEC SX vector supercomputers can achieve high sustained performance in several leading applications. This is due to their high bytes per flop rates compared to scalar systems. However, as recent advantages in VLSI technology have also been accelerating processor speeds, vector supercomputers have been encountering the memory wall problem. As a result, it is getting harder for vector supercomputers to keep a high memory bandwidth balanced with the improvement of their flop/s performance. On the NEC SX systems, the bytes per flop rate has decreased from 8 B/FLOP to 2 B/FLOP during the last twenty years.

* E-mail: musa@sc.isc.tohoku.ac.jp
† E-mail: yoshiei@sc.isc.tohoku.ac.jp
‡ E-mail: egawa@isc.tohoku.ac.jp
§ E-mail: tacky@isc.tohoku.ac.jp
* E-mail: okabe@isc.tohoku.ac.jp
† E-mail: koba@isc.tohoku.ac.jp
We have proposed an on-chip cache, called vector cache, to maintain the effective memory bandwidth rate of future vector supercomputers (Kobayashi et al. (2007)) and provided the early evaluation of the vector cache using the NEC SX simulator (Musa et al. (2007)). The vector cache employs a bypass mechanism between a vector register and a main memory under software controls, and load/store data are selectively cached. The bypass mechanism and the cache work complementarily together to provide data to the register file, resulting in the high effective memory bandwidth rate. However, we did not investigate the fundamental configuration of the vector cache, i.e. cache associativity and cache latency. Moreover, our previous work did not discuss the relationship between loop unrolling and caching. The main contribution of this paper is to clarify the characteristics of the on-chip vector cache on the vector supercomputers. We investigate the relationship among cache associativity, cache latency and performance using five practical applications selected from the areas of advanced computational sciences. In addition, we discuss in detail selective caching and loop unrolling under the vector cache to improve the performance.

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 describes a vector architecture with an on-chip vector cache discussed in this paper. Section 4 provides our experimental methodology and benchmark programs for performance evaluation. Section 5 presents experimental results when executing the kernel loops and five real applications on our architecture. Through the experimental results, we clarify the characteristics of the vector cache. In Section 6, we examine the effect of selective caching on the performance, in which only the data with high localities of reference are cached. In addition, we discuss the relationship between loop unrolling and the vector cache. Finally, Section 7 summarizes the paper with some future work.

2. Related Work

Vector caches have been previously studied by a number of researchers using trace driven simulators of convectional vector architectures in the 1990's. Gee et al. have provided an evaluation of the cache performance in vector architectures: Cray X-MP and Ardent Titan (Gee and Smith (1992), Gee and Smith (1994)). Their caches employ a full-associative cache and an n-way set associative cache. The line sizes range from 16 to 128 bytes, and the maximum cache size is 4 MB. Their benchmark programs are selected from Livermore Loops, NAS kernels, Los Alamos benchmark set and real applications in chemistry, fluid dynamics and linear analysis. They showed that vector references contained somewhat less temporal locality, but large amounts of spatial locality compared to instruction and scalar references, and the cache improved the computational performance of the vector processors. Kontothanassis et al. have evaluated the cache performance of Cray C90 architecture using NAS parallel benchmarks (Kontothanassis et al. (1994)). Their cache is a direct-mapped cache with a 128 bytes line size, and the maximum cache size is 8 MB. They showed that the cache reduced memory traffic from off-chip memory, and a DRAM memory system with the cache was competitive to a SRAM memory system.

Modern vector supercomputer Cray X1 has a 2 MB vector cache, called Ecache, organized as a 2-way set associative write-back cache with 32 bytes line size and the least recently used (LRU) replacement policy (Abts et al. (2003)). The performance of Cray X1 has been evaluated using real scientific applications by several researchers (Dunigan Jr. et al. (2005), Oliker et al. (2004), Shan and Strohmaier (2004)). These studies have compared the performance of Cray X1 with that of other platforms; IBM Power, SGI Altix and NEC SX. However, they have not quantitatively discussed the effect of the cache on its performance.

The aim of our research is to quantitatively investigate the effect of the vector cache using modern vector supercomputer NEC SX architecture. We clarify the basic characteristics of the vector cache and discuss its effective usages.

3. On-Chip Cache Memory for Vector Architecture

We design an on-chip vector cache for a vector processor architecture as shown in Figure 1. Our vector processor has a superscalar unit, a vector unit and an address control unit. The vector unit contains five types of vector arithmetic pipes (Mask, Logical, Multiply, Add/Shift and Divide), vector registers and a vector cache. Four parallel vector pipes

![Figure 1. Vector architecture with vector cache and memory system block diagram.](image-url)
are provided for each type; total 20 pipes are included in the vector unit. The address control unit queues and issues vector operation instructions to the vector unit. The vector operation instructions contain vector memory instructions and vector computational instructions. Each instruction concurrently deals with 256 double-precision floating-point data.

The main memory unit employs an interleaved memory system. The main memory is divided into 32 parts, each of which operates independently and supplies data to the vector processor. Thus, the main memory and the vector processor are interconnected through 32 memory ports. Moreover, we implement the vector cache in each memory part; the vector cache consists of 32 sub-caches, and the sub-cache is a non-blocking cache with a bypass mechanism between a vector register file and the main memory. The bypass mechanism is software-controlled, and vector load/store instructions have a flag for cache control, i.e. cache on/off. When the flag indicates “cache on,” the supplied data by the vector load/store instruction are cached. Meanwhile, when the flag is “cache off,” the vector load/store instruction provides data via the bypass mechanism: the data are not cached. The sub-cache employs a set-associative write-through cache with the LRU replacement policy. The line size is 8 bytes; it is the unit for memory accesses.

4. Experimental Environment

4.1 Methodology

We use a trace-driven simulator that can simulate the behavior of the proposed vector architecture at the register transfer level. It is based on the NEC SX simulator, which accurately models a single processor of the SX architecture; the vector unit, the scalar unit and the memory system. The simulator takes a system parameter file and a trace file as input, and the output of the simulator contains instruction cycle counts of a benchmark program and cache hit information. The system parameter file has configuration parameters of a vector architecture and setting parameters, i.e., the cache size, the associativity, the cache latency and the memory bandwidth.

The trace file contains an instruction sequence of a benchmark program and directives of cache control: cache on/off. These directives are used in selective caching and set the cache control flags in the vector load/store instructions. A benchmark program is compiled by the NEC FORTRAN compiler: FORTRAN90/SX. It supports ANSI/ISO Fortran95 in addition to functions of automatic vectorization and automatic parallelization. The executable program runs on the SX trace generator to produce the trace file. In this work, we use two kernel loops and the original source codes of five applications, and compile them with the highest optimizations option (-C hopt) and inlining subroutines.

To evaluate on-chip vector caching for future vector processors with a higher flop/s rate but a relatively lower off-chip memory bandwidth, we evaluate the effect of the cache on the vector processor by limiting its memory bandwidth per flop/s rate from 4 B/FLOP down to 1 B/FLOP. Here, we adopt the NEC SX-7 architecture (Kitagawa et al. (2003)) to our vector processor. The setting parameters are shown in Table 1.

<table>
<thead>
<tr>
<th>Table 1. Summary of setting parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base System Architecture</strong></td>
</tr>
<tr>
<td><strong>Main Memory</strong></td>
</tr>
<tr>
<td><strong>Vector Cache</strong></td>
</tr>
<tr>
<td><strong>Total Size (Sub-cache Size)</strong></td>
</tr>
<tr>
<td><strong>Cache Policy</strong></td>
</tr>
<tr>
<td><strong>Associativity</strong></td>
</tr>
<tr>
<td><strong>Cache Latency</strong></td>
</tr>
<tr>
<td><strong>Cache Bank Cycle</strong></td>
</tr>
<tr>
<td><strong>Line Size</strong></td>
</tr>
<tr>
<td><strong>Memory—Cache bandwidth per flop/s</strong></td>
</tr>
<tr>
<td><strong>Cache—Register bandwidth per flop/s</strong></td>
</tr>
</tbody>
</table>

4.2 Benchmark Programs

To clarify the basic characteristics and validity of the vector cache, we select basic kernel loops and five leading applications. In this section, we describe our benchmark programs in detail.

The following kernel loops are used to evaluate the performance of the vector processor with the cache.

\[
A(i) = X(i) + Y(i),
\]

\[
A(i) = X(i) \times Y(i) + Z(i).
\]
As leading applications, we have selected five applications in three scientific computing areas. The applications have been developed by researchers of Tohoku University for the SX-7 system and are representative of each research area. Table 2 shows the summary of the evaluated applications, which methods are standard in individual areas (Kobayashi (2006)).

Table 2. Summary of benchmark programs.

<table>
<thead>
<tr>
<th>Area</th>
<th>Name and Description</th>
<th>Method Subdivision</th>
<th>Memory Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electromagnetic Analysis</td>
<td>GPR simulation: Simulation of Array Antenna Ground Penetrating Radar</td>
<td>FDTD 50 × 750 × 750</td>
<td>8.9 GB</td>
</tr>
<tr>
<td></td>
<td>APFA simulation: Simulation of Anti-Podal Fermi Antenna</td>
<td>FDTD 612 × 105 × 505</td>
<td>12 GB</td>
</tr>
<tr>
<td>CFD/Heat Analysis</td>
<td>PRF simulation: Simulation of Premixed Reactive Flow in Combustion</td>
<td>DNS 513 × 513</td>
<td>1.4 GB</td>
</tr>
<tr>
<td></td>
<td>SFHT simulation: Simulation of Separated Flow and Heat Transfer</td>
<td>SMAC 711 × 91 × 221</td>
<td>6.6 GB</td>
</tr>
<tr>
<td>Seismology</td>
<td>PBM simulation: Simulation of Plate Boundary Model on Seismic Slow Slip</td>
<td>Friction Law 32400 × 32400</td>
<td>8 GB</td>
</tr>
</tbody>
</table>

The GPR simulation code evaluates a performance of the SAR-GPR (Synthetic Aperture Radar–Ground Penetrating Radar) in detection of buried anti personnel mines under conditions of inhomogeneous subsurface mediums (Kobayashi et al. (2003)). The simulation method is based on the three dimensional FDTD (Finite Difference Time Domain) method (Kunz and Luebbers (1993)). The vector operation ratio is 99.7% and the vector loop length is over 500.

The APFA simulation is applied to design of a high gain antenna; an Anti-Podal Fermi Antenna (APFA) (Takagi et al. (2004)). The simulation calculates directional characteristics of the APFA using the FDTD method and the Fourier transform. The vector operation ratio is 99.9% and the vector loop length is 255.

The PRF simulation provides numerical simulations of two dimensional Premixed Reactive Flow (PRF) in combustion for design of engines of planes (Tsuboi and Masuya (2003)). The simulation uses the sixth-order compact finite difference scheme and the third-order Runge-Kutta method for time advancement. The vector operation ratio is 99.3% and the vector loop length is 513.

The SFHT simulation realizes direct numerical simulations of three-dimensional laminar Separated Flow and Heat Transfer (SFHT) on surfaces of a plane (Nakajima et al. (2005)). The finite-difference forms are the fifth-order upwind difference scheme for space derivatives and the Crank–Nicholson method for a time derivative. The resulting finite-difference equations are solved using the SMAC method. The vector operation ratio is 99.4% and the vector loop length is 349.

The PBM simulation uses the three-dimensional numerical Plate Boundary Models (PBM) to explain an observed variation in propagation speed of postseismic slip (Ariyoshi et al. (2007)). This is a quasi-static simulation in an elastic half-space including a rate- and state-dependent friction. The vector operation ratio is 99.5% and the vector loop length is 32400.

5. Performance Evaluation of Vector Cache

We evaluate the performance of the proposed vector processor architecture by using the benchmark programs and clarify the basic characteristics of the vector cache.

5.1 Impact of Memory Bandwidth on Effective Performance

To evaluate the impact of the memory bandwidth on performance of the vector processor without the vector cache, we varied memory bandwidth per flop/s rates from 4 B/FLOP down to 1 B/FLOP. Figure 2 shows the computation efficiency, the ratio of the sustained performance to the peak performance, in the execution of the benchmark programs. Here, the 4 B/FLOP case is equal to the memory bandwidth per flop/s rate of the SX-8 or earlier system, the 2 B/FLOP is the same as Cary X1 and BlackWidow (Abts et al. (2007)), and the 1 B/FLOP is the same level as the commodity-based high performance scalar systems such as NEC SX series (Sen D. et al. (2003)) and SGI Altix series (SGI (2007)). Figure 2 indicates that the computational performance is seriously affected by the B/FLOP rate. In particular, the performances of GPR and PBM, which are memory-intensive programs, are degraded by half and quarter when the memory bandwidth per flop/s rate is reduced to 2 B/FLOP and 1 B/FLOP from 4 B/FLOP. Even if APFA is computation-intensive, the performance on the 1 B/FLOP system is decreased to 69% of the performance on the 4 B/FLOP system. Therefore, the memory bandwidth per flop/s rate needs the 4 B/FLOP rate on the vector supercomputer.
5.2 Relationship between Efficiency and Cache Hit Rate on Kernel Loops

We simulate the execution of the kernel loop (1) on the vector processor with the vector cache. Since this loop has no temporal locality, we store a part of $X(i)$ and $Y(i)$ data on the vector cache in advance of the loop execution. In the following two ways, we select the data for caching to change the range of cache hit rates, and examine their effects on performance. One is to store both $X(i)$ and $Y(i)$ with the same index $i$ on the cache, and load instructions of both $X(i)$ and $Y(i)$ are set to "cache on," named Case 1. The other is to cache only $X(i)$, and load instructions of $X(i)$ and $Y(i)$ are set to "cache on" and "cache off," respectively. This is Case 2. Here, the range of index $i$ is $1 \leq i \leq 25600$.

Figure 3 shows the relationship between the cache hit rate and the relative memory bandwidth that is obtained by normalizing the effective memory bandwidth of the systems with the vector cache by that of the 4 B/FLOP system without the cache. Figure 4 shows the relationship between the cache hit rate and the computation efficiency. Figure 3 indicates that the relative memory bandwidth of each case increases as the cache hit rate increases. Therefore, the vector cache is one of the promising solutions to cover a lack of the memory bandwidth. Similar to Figure 3, the computation efficiency of each case in Figure 4 improves as the cache hit rate increases.

Figure 3. Relationship between cache hit rate and relative memory bandwidth on Kernel loop (1).

Figures 3 and 4 show the correlation between the relative memory bandwidth and the computation efficiency. On both the 2 B/FLOP and 1 B/FLOP systems, the performance of Case 2 is greater than that of Case 1. In particular, Case 2 with the vector cache of the 50% hit rate on the 2 B/FLOP system achieves the same performance of the 4 B/FLOP system. On the 4 B/FLOP system, each of $X(i)$ and $Y(i)$ is provided at a 2 B/FLOP bandwidth rate on average. When the cache hit rate is 50% in Case 2, all data of $X(i)$ are supplied directly from the vector cache and all data of $Y(i)$ are supplied from the memory at the 2 B/FLOP bandwidth rate through the bypass mechanism. Consequently, each of $X(i)$ and $Y(i)$ is provided at the 2 B/FLOP bandwidth rate on the vector registers, and the total amount of data provided to vector registers in time is equal to that of the 4 B/FLOP system. However, the 1 B/FLOP system with vector caching with the 50% hit rate does not achieve the performance of the 4 B/FLOP system. Because $Y(i)$ is provided at a 1 B/FLOP bandwidth rate, the total bandwidth rate at the vector registers does not reach the 4 B/FLOP. On the other hand, in Case 1, the data of $X(i)$ and $Y(i)$ are supplied from either of the vector cache or the memory at the same index $i$. While supplying the data from the memory, $X(i)$ and $Y(i)$ have to share the memory bandwidth rate. Therefore, the 1 B/
FLOP and 2 B/FLOP systems need a cache hit rate of 100% to achieve a performance comparable to that of the 4 B/FLOP system.

Similarly, we examine the performance of Kernel (2) on the 2 B/FLOP system in the following three cases. In Case 1, all of \(X(i), Y(i),\) and \(Z(i)\) are cached with the same index \(i\) in advance. In Case 2, both \(X(i)\) and \(Y(i)\) are provided from the vector cache, and therefore the maximum cache hit rate is 66%. In Case 3, only \(X(i)\) is in the cache, and hence the maximum cache hit rate is 33%. Figure 5 shows the relationship between the cache hit rate and the change in the relative memory bandwidth regarding Kernel (2). On the 4 B/FLOP system, each of \(X(i), Y(i),\) and \(Z(i)\) is provided at a 4/3 B/FLOP bandwidth rate on average. The performance of Case 2 is comparable to that of the 4 B/FLOP system when the cache hit rate is 66%, because the bandwidth per flop/s rate of each data is the 4/3 B/FLOP rate on average. However, in Case 3, both \(Y(i)\) and \(Z(i)\) are provided from the memory, and \(Y(i)\) and \(Z(i)\) have to share the 2 B/FLOP bandwidth rate at the vector register. As a result, the relative memory bandwidth never reaches that of the 4 B/FLOP system.

These results indicate that the vector cache has a potential for the future SX systems to cover the shortage of memory bandwidth. In additions, all data do not need to be cached from the discussions on Figures 3 and 5. The key data determining the performance should be cached to make good use of limited on-chip capacity.

### 5.3 Relationship between Efficiency and Cache Hit Rate on Five Scientific Applications

We simulate the execution of five scientific applications with the vector cache varying memory bandwidth per flop/s rate; 1 B/FLOP and 2 B/FLOP. Here, all vector load/store instructions are set to “cache on.” Figure 6 shows the relationship between the cache hit rate of the 8 MB vector cache and the relative memory bandwidth. Figure 6 indicates that the vector cache can improve the effective memory bandwidth, depending on the cache hit rate of the applications. Cache hit rates vary from 13% in SFHT to 96% in APFA, because these depend on the locality of reference in individual applications.

In APFA, the relative memory bandwidth is 0.96 at the 2 B/FLOP system with almost 100% hit of the 8 MB cache, resulting in an effective memory bandwidth almost equal to the 4 B/FLOP system. In this case, most data are provided...
from the cache. PBM of the 2 B/FLOP system reaches the effective memory bandwidth of the 4 B/FLOP system at the cache hit rate 50%. Figure 7 shows the main routine of PBM. The inner loop, index \( j \), is vectorized and arrays \( gd\_dip \) and \( wary \) are stored in the vector cache by vector load instructions. However, \( gd\_dip \) is spilled from the cache, because \( gd\_dip \) needs 7.6 GB. On the other hand, \( wary \) can be held in the cache, because its needs only 250 KB and the array is defined in the preceding loop. Therefore, the cache hit rate of this loop is 50%. Then all data of \( wary \) are supplied directly from the vector cache at the 2 B/FLOP rate and all data of \( gd\_dip \) are supplied from the memory at the 2 B/FLOP rate through the cache, and the total amount of data provided to vector registers in time is equal to that of the 4 B/FLOP system.

The relative memory bandwidths in other applications, SFHT, GPR, and PRF are over 0.6 on the 2 B/FLOP system and 0.3 on the 1 B/FLOP system. Here, the relative memory bandwidth without the cache is 0.5 on the 2 B/FLOP system and 0.25 on the 1 B/FLOP system. Thus, the improvement of the relative memory bandwidth is over 20% using the 8 MB cache.

Figure 8 shows the computation efficiency of five scientific applications on the 4 B/FLOP system without the cache, the 2 B/FLOP and 1 B/FLOP systems with/without the 8 MB cache. Figure 8 indicates that the efficiency of the 2 B/FLOP and 1 B/FLOP systems is increased by the cache. In particular, the 2 B/FLOP system with the cache achieves the same efficiency as the 4 B/FLOP system in APFA and PBM. Figure 9 shows the relationship between the cache hit rate of the 8 MB cache and the recovery rate of the performance by the cache from the 2 B/FLOP and 1 B/FLOP systems to the 4 B/FLOP system. The recovery rate goes up as the cache hit rate increases. The vector cache can increase the recovery rate by 18 to 99% on the 2 B/FLOP system and 9 to 96% on the 1 B/FLOP system, depending on the data access locality of each application. The results indicate that the vector cache has a potential for the future SX systems to cover the shortage of memory bandwidth on real scientific applications.

### 5.4 Relationship between Associativity and Cache Hit Rate

In this section, we examine the effect of the associativity ranging from 2 to 8 ways on the performance. Figure 10 shows that the cache hit rate on each set associative cache, covering cache sizes from 256 KB to 8 MB. Four applications, APFA, PRF, SFHT, and PBM, approximately have constant cache hit rates across three associativity cases. In GPR, the cache hit rate slightly varies with the associativity.

The cache hit rate is dependent on among placements of the arrays on memory addresses, memory access patterns of the programs and the associativity. In GPR, its basic computational structure consists of triple nested loops. The loops access many arrays at intervals of 584 bytes stride addresses, and the cache data are frequently replaced. Therefore, the
cache hit rate varies with the cache capacities and the associativity. In SFHT, the memory access patterns have 8 bytes stride accesses. On the 2 MB case, the cached data which will be reused on the 2-way set associative cache are evicted by some other data. Thus this is due to a placement of memory addresses of the arrays. Other programs have constant cache hit rates across three associativity cases. These memory access patterns are 8 bytes or 16 bytes stride accesses, and the placement of the arrays on memory do not cause the conflict of reused data.

These results provide that the associativity hardly have effects on the cache hit rates, and the cache capacity mainly influences the cache hit rate across the five applications.

5.5 Effects of Vector Cache Latency on Performance

In general, the latency of the on-chip cache is considerably shorter than that of the off-chip memory. We investigate the effects of the cache access latency on the performance of the five applications. Figure 11 shows the relationship between the computation efficiency of the applications and four cache latencies; 15, 50, 85 and 100% of the memory access latency on the 2 B/FLOP system with the 2 MB cache. The five applications have almost the same efficiency when changing the cache latency. Because the vector architecture can hide the memory access times by pipelined vector operations when a vector operation ratio and a vector loop length of applications are enough large. In addition, the vector cache consists of 32 sub-caches between a vector register file and the main memory via each memory port. The sub-caches connected to 32 memory ports provide multiple words at a time. The cache latency of the second memory access or later is hidden in the sub-cache system.

For comparison, we examine the performance of shorter loop length cases using Kernel loops (1) and (2) on the 2 B/FLOP system when changing the cache access latency. Figure 12 shows that the relationship between the computation efficiency of Kernel loops and four cache latency cases. Here, we examine three loop length cases in the 100% cache hit rate; $1 \leq i \leq 64$, $1 \leq i \leq 128$, $1 \leq i \leq 256$. On the shorter loop length cases, Figure 12 indicates that the vector cache latency is an importance factor in the performance. Especially, on the loop length 64 case, the 15% latency case has two times higher efficiency in Kernel (1) than the 100% case, and the 15% latency case has 12% higher efficiency in Kernel
However, the longer the loop length, the lower the effect of the cache latency. Figure 13 shows that the computation efficiency of Kernel loops in the 4B/FLOP and 2B/FLOP systems with/without the vector cache. On both the loop lengths 64 and 128 of Kernel (1), the efficiency of the 4B/FLOP system with the cache is higher than that of the 4B/FLOP system without the cache, and the loop length 128 case of the 4B/FLOP system with the cache achieves the highest efficiency. Meanwhile, in Kernel (2), the loop length 64 case of the 4B/FLOP system with the cache has the highest efficiency. This is because the ratio of memory access time to the processing time becomes the

![Figure 10. Cache hit rate vs. associativity on five applications.](image)

![Figure 11. Computation efficiency of five applications on four cache latency cases.](image)
smallest due to the decrease in memory latency by the cache, and in these kernel loops the memory access time of the shorter loop length case cannot be entirely hidden by pipelined vector operations only. Therefore, the performance of shorter loop length cases is sensitive to the vector cache latency, and the vector cache can boost the performance of the applications with short vector lengths on the 4 B/FLOP system.

Moreover, the cache latency is not hidden when the bank conflicts occur. In this work, the bank conflicts hardly occur in the five applications. As our future work, quantitative discussions on the influence of the bank conflicts will be addressed.

6. Optimizations for Vector Caching: Selective Caching and Loop Unrolling

In this section, we discuss some optimization techniques for the vector cache to reduce cache miss rates on scientific applications. In particular, we show that the relationship between loop-unrolling and caching. Many compilers automatically optimize programs using the loop-unrolling, which is the most basic loop optimization.

6.1 Effects of Selective Caching

The vector cache has a bypass mechanism, and data are selectively cached: selective caching. The simulation methods of the four scientific applications, GPR, APFA, PRF and SFHT, are based on the difference schemes; Finite Difference Time Domain (FDTD) method and Direct Numerical Simulation of flow and heart (DNS), and a part of arrays has a high locality in many cases. As the on-chip cache size is limited, selective caching might be an effective approach to efficient use of the on-chip cache. We evaluate the selective caching for two applications, GPR and PRF, which are typical examples of FDTD and DNS, respectively.

Figure 14 shows one of kernel loops of GPR; the loop is a main routine of the FDTD method, and many routines of GPR are similar to this kernel loop. The innermost loop is indexed with \( j \). Usually, FORTRAN compilers interchange loops \( j \) and \( i \), however, FORTRAN90/SX compiler does not interchange the loops, because the compiler decides that the computational performance cannot be improved due to the short length of loop \( i \). Loop \( i \) has a length of 50, and \( j \) is 750.

As the size of each array used in GPR is 330 MB, the cache cannot store all the arrays. However, the difference scheme as shown in Figure 14 generally has temporal locality in accessing arrays; \( H_x, H_y \) and \( H_z \). We select these arrays for selective caching. Figure 15 shows the effect of selective caching on the 2 B/FLOP system with the vector cache in GPR. Figure 16 shows the recovery rate of the performance from the 2 B/FLOP system without the cache to the 4 B/FLOP system by selective caching in GPR. We discuss the cache sizes from 256 KB up to 8 MB. Here, “ALL” in the figures indicates that all the arrays in the loop are cached. “Selective” shows that some of the arrays are selectively cached. “Cache Usage Rate” indicates the ratio of number of cache hit references to the total memory references.
In the 256 KB cache case, the efficiencies of “ALL” and “Selective” are 33.3 and 34.1%, and the cache usage rate are 9.6 and 16.6%, respectively. The arrays $H_y(i,j,k)$, $H_y(i-1,j,k)$ (line 08 in Figure 14), $H_x(i,j,k)$ (line 11) and $H_z(i-1,j,k)$ (line 12) can load data from the cache on “Selective.” In “ALL,” $H_y(i-1,j,k)$ (line 08) and
$H_z(i-1,j,k)$ (line 12) cause cache misses, because the reused data of these arrays are replaced by other array data. Figures 15 and 16 indicate that the performance and the cache usage rate increase as the cache size increase. On the 4 MB cache, the cache hit rate of “Selective” is 24% and its efficiency is 3% higher than that of “ALL.” The recovery rate of performance is mere 34%. However, GPR cannot achieve high cache usage rates and recovery rates of performance by selective caching. This is because this loop has many non-locality arrays; $C_{x,a}$, $E_{x}$, $E_{x,Current}$ etc.

01 DO $KK = 2,NJ$
02 DO $I = 1,NI$
03  $wDY1YC3(I,KK-,1,L) = wDY1YC3(I,KK-,1,L) \ast wDY1YC2(I,KK-,1,L)$
04  $wDY1YC2(I,KK,L) = wDY1YC2(I,KK,L) - wDY1YC1(I,KK,L) \ast wDY1YC3(I,KK-,1,L)$
05  $wDY1YC2(I,KK,L) = 1.00 / wDY1YC2(I,KK,L)$
06  $wDY1YC(I,KK,L) = (wPHIY12(I,KK) - wDY1YC1(I,KK,L) \ast wDY1YC1(I,KK-,1,L)) \ast wDY1YC2(I,KK,L)$
07  END DO
08  END DO

Figure 17. High cost kernel loop of PRF.

Figure 17 shows one of kernel loops of PRF; the loop is a high cost routine. The size of each array in PRF is 18 MB, and many loops are doubly nested loop. Thus, the arrays are treated as two dimensions array, and the size of cached data per array is 2 MB only. $wDY1YC1$, $wDY1YC2$ and $wDY1YC3$ in Figure 17 are defined in the preceding loop. In addition, $wDY1YC1$ and $wDY1YC2$ have the spatial locality. We select these arrays for selective caching. Here, $wDY1YC3(I,KK-,1,L)$ (line 04 in Figure 17), $wDY1YC2(I,KK,L)$ (line 05), $wDY1YC1(I,KK,L)$ (line 06) and $wDY1YC2(I,KK,L)$ (line 06) can reuse data on the register files. Thus, these arrays do not access the cache.

![Figure 18. Efficiency and cache hit rate of selective caching on performance (PRF).](image)

Figures 18 and 19 show the efficiency and the recovery rate of the performance from the 2 B/FLOP system to the 4 B/FLOP system by selective caching in PRF. Figure 18 indicates that the cache hit rates and efficiency of “Selective” are the same as that of “ALL” until 2 MB cache size. $wDY1YC2(I,KK-,1,L)$ (line 03) and $wDY1YC1(I,KK-,1,L)$ (line 08) are cache hit arrays in each case, and the cache hit rate is 33%. Over 4 MB cache size, “Selective” achieves a 66% cache hit rate, resulting in a 30+% improvement in efficiency. Moreover, Figure 19 indicates that the recovery rate is 78%.

The results of GPR and PRF show that the selective caching improves the performance of the applications. Compared with GPR, PRF has higher values regarding both the cache usage rate and improved efficiency, because the ratio of arrays with locality per loop on PRF is higher than that of GPR. In GPR the reused data are only defined in the loop, and the cache miss always occurs in this loop. For example, arrays $H_{x}(i,j,k)$ (line 08 in Figure 14) hits the data of $H_{x}(i,j,k)$ (line 06), but $H_{x}(i,j,k)$ (line 06) cannot hit the data due to their first accesses (cold miss). On the other hand, the cached data of PRF are provided in the immediately preceding loop, thus cache misses do not occur like the first access in PRF.
6.2 Effects of Loop-unrolling and Caching

Loop unrolling is effective for higher utilization of vector pipelines, however, loop unrolling also needs more cache capacity to capture all the arrays of unrolled loops. Therefore, we discuss the relationship between loop unrolling and vector caching on the performance.

Figure 20 shows a code outer unrolled 4 times, whose original code is shown in Figure 7. In the outer unrolling case, array \textit{wary} of the outer loop index \textit{i} (line 03 in Figure 20) reuses the data of the index \textit{i} – 1 on the cache and arrays \textit{wary} (line 04, 05 and 06) reuse the data on register files of \textit{wary} (line 03), hence, memory references are reduced. However, the cache capacity for \textit{gd dip} increases due to an increase in the number of arrays reference in the innermost loop. Figure 21 shows the efficiency and the cache hit rate of the outer unrolling case on PBM with the 2 B/FLOP system. “2B/F” indicates the without-cache case, and the efficiency constantly increases as the degree of unrolling increases. The cache hit rate of the 0.5 MB cache case is 49% on the non-unrolling case and becomes poor as the loop unrolling proceeds, because the cached data of \textit{wary} are evicted from the cache by \textit{gd dip}. Thus, the efficiency of the 0.5 MB cache case achieves 49% on the non-unrolling case, and it is similar to that of the 2B/FLOP system with unrolling. In this case, a conflicting effect between caching and unrolling appears. Moreover, the cache hit rates of the 8 MB cache case decrease as the degree of unrolling increases. The cached data remain on the cache in this case, but the cache hit rate decreases due to a decrease in the number of \textit{wary} (line 03) references. However, the efficiency is approximately constant; 48 or 49% because unrolling covers the losing effect of caching. This case shows that the effects of caching are comparable to that of unrolling.

Figure 22 shows a code obtained by unrolling code two times shown in Figure 17. Outer unrolling reduces the memory references of the arrays \textit{wDYYC2(I,KK,L)} (line 07 in Figure 22) and \textit{wDYYC1(I,KK,L)} (line 10). Figure 23 shows the efficiency and the cache hit rate of outer unrolling on PRF with the 2B/FLOP system. The cache hit rates of both the 0.5 and 8 MB caches gradually reduce as the degree of unrolling increase, because of an increase in the cache miss rate. In this case, the highest efficiency is 33% on the 8 MB cache with the unrolling degree of 4 since the gain by loop unrolling covers the loss due to the increase in the miss rate. It is higher than the selective caching case; it is 30.7% (Ref. Figure 18). Thus, this case indicates the synergistic effect of both caching and unrolling.
These results indicate that loop unrolling has both conflicting and synergistic effects of caching. Loop unrolling is a most basic optimization and has beneficial effects in various applications. Therefore, caching has to be used carefully as a complementary tuning option for loop unrolling.

7. Conclusions

This paper has presented the characteristics of an on-chip vector cache with the bypass mechanism on the vector supercomputer NEC SX architecture. We have demonstrated the relationship between the cache hit rate and the
Characteristics of an On-Chip Cache on NEC SX Vector Architecture

performance using two DAXPY-like loops. Moreover, we have clarified the same tendency of the relationship on the five scientific applications. The vector cache can recover the lack of the memory bandwidth, and boosts the computation efficiency of the 2 B/FLOP and 1 B/FLOP systems. Especially, as cache hit rates are 50% or more, the 2 B/FLOP system can achieve a performance comparable to the 4 B/FLOP system. The vector cache with a bypass mechanism can provide the data from both the memory and the cache in time, and the sustained memory bandwidth for the registers is increased. Therefore, the vector cache has a potential for the future SX systems to cover the shortage of their memory bandwidth.

We have also discussed the relationship between performance and cache design parameters such as cache associativity and cache latency. We have examined the effect of the cache associativity setting from 2-way to 8-way. When the associativity is two or more, the cache hit rate is not sensitive to the associativity in the five applications. In addition, we have examined the effect of the cache latency on the performance when changing it to 15, 50, 85 and 100% of the memory latency. We have demonstrated the computational efficiencies of five applications are constant across these latency changes, when the vector loop lengths of the applications are 256 or more. In these cases, the latency is hidden by pipelined vector operations. However, in the case of shorter vector loop lengths, the cache latency affects the performance, and the 15% latency case of Kernel (1) has two times higher efficiency than the 100% case in the 2 B/FLOP system. In addition, the 4 B/FLOP system is also boosted due to the effect of the short latency of the cache. The above results indicate that the performance of cache does not largely depend on the cache associativity, however, the cache latency affects the performance of applications with short vector loop lengths.

Finally we have discussed selective caching and the relationship between loop-unrolling and caching. We have shown that selective caching, which is controlled by means of the bypass mechanism, is effective for efficient use of limited on-chip caches. We have examined two cases; the ratios of arrays with locality per loop are higher and lower cases. In each case, the higher performance is obtained by selective caching, compared with all the data caching. In addition, the loop unrolling is useful in the improvement of performance, and caching is complementary to the effect of loop unrolling.

In the future work, we will evaluate other scientific applications using the vector cache, and quantitatively discuss the relationship between the bank conflicts and the cache latency. In addition, we will also investigate the mechanism of vector cache on multi core vector processors.

Acknowledgments

The authors would like to gratefully thank: Dr. M. Sato, Dr. A. Hasegawa, Professor G. Masuya, Professor T. Ota, Professor K. Sawaya of Tohoku University, for their advice about applications. We also would like to thank Matsu Aoyama and Hirofumi Akiba of NEC Software Tohoku for their assistance in experiments.

REFERENCES


Aoyama and Hirofumi Akiba of NEC Software Tohoku for their assistance in experiments.

Professor K. Sawaya of Tohoku University, for their advice about applications. We also would like to thank Matsuo Aoyama and Hirofumi Akiba of NEC Software Tohoku for their assistance in experiments.


