Memory Bandwidth: The True Bottleneck of SIMD Multimedia Performance on a Superscalar Processor

Julien Sebot and Nathalie Drach-Temam

LRI - Université Paris Sud - France

Abstract. This paper presents the performance of DSP, image and 3D applications on recent general-purpose microprocessors using streaming SIMD ISA extensions (integer and floating point). The 9 benchmarks benchmark we use for this evaluation have been optimized for DLP and caches use with SIMD extensions and data prefetch. The result of these cumulated optimizations is a speedup that ranges from 1.9 to 7.1.
All the benchmarks were originaly computation bound and 7 becomes memory bandwidth bound with the addition of SIMD and data prefetch. Quadrupling the memory bandwidth has no effect on original kernels but improves the performance of SIMD kernels by 15-55%.

1 Introduction

The SIMD (Single Instruction Multiple Data) extensions to general purpose microprocessor instruction sets have been introduced in 1995 to enhance the performance of multimedia and DSP applications. The goal for these extensions to general purpose microprocessors instruction sets is to accelerate DSP and multimedia application at a small cost. Traditionally in these processors, better ILP exploitation goes through increasing dispatch logic (reorder buffer size, reservation station size) and the number of functional units available. With SIMD instruction sets only the number of functional units is increased (the width of a functional unit). With constantly increasing available transistors in microprocessors there was room for the additional functional units needed by SIMD instruction sets. With SIMD extensions it's up to the programmer or the compiler to extract the ILP that is traditionally extracted by the out of order core of the microprocessor. Multimedia and DSP applications generally work on data vectors, so SIMD programmation is well adapted to these classes of applications.

At the beginning, these extensions were able to process 64 bit integer vectors: Sun with VIS in 1995, HP with MAX in 1995, MIPS with MDMX, Intel with MMX in 1997. The introduction of SIMD extensions able to process single precision floating-point vectors is more recent: AMD with 3DNow! [2] and Intel with SSE in 1998. The multimedia applications use more and more floating-point computations. Currently the main target for these FP SIMD extensions is 3D polygonal rendering that is extensively used in CAD applications and games. The need for single precision floating-point computing power is very important in

R. Sakellariou et al. (Eds.): Euro-Par 2001, LNCS 2150, pp. 439–447, 2001.

this class of applications. In 1999 Motorola introduced AltiVec, which is the first SIMD instruction set that can handle both integer and floating-point vectors. Recently, Intel introduced SSE2 with Pentium 4, SSE2 enables double precision floating-point SIMD computation.

In this paper we study the performance of both integer and floating-point applications through DSP, image and 3D kernels on a general purpose microprocessor with SIMD extensions. DSP, image and 3D are the three main classes of multimedia applications and multimedia is the main workload on home computers. We show that SIMD optimizations and data prefetching raise the performance by 1.9 to 7.1 and that memory bandwidth becomes the bottleneck.

2 Related Work and Motivation

This section discusses related work in SIMD instruction sets evaluation.

In [7], Ranganathan and Al. study the performance of several integer image processing kernel including microbenchmarks, jpeg and mpeg compression. They show improvements from 1.1x to 4.2x with VIS, and that data prefetching is very efficient on media kernels. In [1] Bhagarva and Al. show improvements from 1.0x to 6.1x on DSP kernels and jpeg, but don't evaluate the impact of memory hierarchy on the final performance. Yang and Al. study in [3] the impact of paired SIMD floating-point instructions and 4-way SIMD floating-point instructions on 3D geometry transformations. Their study is limited to the impact on ILP and does not look at the memory hierarchy problems. In [6] Nguyen and Al. study the ILP of a small number of micro kernels with AltiVec and show improvements without memory hierarchy between 1.6x and 11.66x.

We have shown preliminary results in [4] about AltiVec performance on floating-point multimedia kernels. In this paper we show the performance improvement of 9 Integer and Floating-Point micro kernels using AltiVec. We focus on the impact of memory hierarchy (latency and bandwidth) on the performance of multimedia kernels using a SIMD instruction set. We show that with AltiVec streaming prefetch enabled, the kernel performance becomes bounded by the memory bandwidth.

3 Methodology

AltiVec is a Single Instruction Multiple Data (SIMD) machine comprised of $32 \times 128 - bit$ "vector" registers, 2 fully pipelined processing units (a vector permute and an ALU) and 170 new instructions. These instructions comprise vector multiply accumulation, all kinds of permutations between two vectors, float to int conversions, pixel packing, logical operations, comparisons, selections and extensions from a scalar to a vector.

AltiVec vector registers can hold either 16 8-bit integers, 8 16-bit integers, 4 32-bit integers than 4 IEEE-754 single precision floats. AltiVec provides streaming prefetch instructions. The programmer can use these instructions to specify the amount of data to be prefetched into the first level cache, with an optional

stride. The Motorola PowerPC 7400 is the first microprocessor to implement AltiVec instruction set. There is more informations about the PowerPC 7400 in [5].

We have used Apple MrC compiler through a Metrowerks Codewarrior pro 4 plug-in. MrC allows easy Altivec programming through some extensions which enable the use of Altivec functions in a C program. The C programming model for Altivec used in MrC is designed to look like function calls, but each of this function call represents and generates one Altivec instruction. Variables are defined using C types that match Altivec data types and variables declared using these types are all 16-byte aligned.

We have evaluated the performance of our multimedia kernel by execution on a real PowerPC G4. We have used trace driven simulation to estimate the impact of memory bandwidth impact on these kernel's performance. We couldn't modify the memory bandwidth of our PowerPC G4 (by changing the bus/processor ratio and the memory latency) so we used Motorola G4 simulator v1.1.2 and pitsTT6 v1.4 execution trace generator. PitsTT6 works as a shared library. A call to *startTrace()* launch the trace generation during the execution and a call to *stopTrace()* stops it. It generates a TT6 execution trace which we used as input for the G4 simulator.

4 SIMD Optimizations for DSP Image and 3D Kernels

We have used 9 micro-kernels taken from multimedia and DSP applications. There are 4 single precision floating-point kernels and 5 integer kernels. SOAD: Sum Of Absolute Difference kernel is found in several reference video coder implementations such as MPEG2. It is one of the most computationally intensive aspects of video encoding. RGB2YCbCr converts from Gamma Corrected RGB to the YCbCr Color Space. It is used in video compression algorithms. iDCT: The Inverse Discrete Cosine Transform is used in MPEG2 decompression. The Image filters corresponds to gamma correction filters on static images. The sparse one works only on one component of the image. FIR and IIR filtering are two of the fundamental operations in digital signal processing. DAXPY is used in number of numerical algorithms, it is the heart of matrix-matrix products. The Max micro-kernel is used to find the maximum floating-point element of a vector and the elements corresponding index. It is a very important step in the Viterbi algorithm (speech recognition). The 3D kernel performs the essential steps of the geometry part of the 3D polygonal graphic pipeline. 3D polygonal rendering is used by most of the CAO/CAD applications, virtual reality and many games.

AltiVec provides SIMD instructions able to work on 4 to 16 element wide vectors depending on the data types. The programmer can permute each byte element of a vector as he wants. A simple way to characterize the "theoritical" speedup of a SIMD program is to use as speedup the vector width in number of elements. It also gives a good idea of the potential instruction count reduction factor. The speedups won't be so high for the following reasons:

– data need often to be reorganized to be processed efficiently with the SIMD instructions available. This overhead limits the speedup of all the programs that need data reorganization.
– The ILP that can be extracted from each algorithm is variable and often the theoretical speedup isn't reachable because of the instructions dependency graph that limits the ILP.

SIMD ISA provide generally specialized instructions that accelerate much the processing of most DSP and multimedia algorithms. These instructions, if they are only present in the SIMD extensions can raise the speedup of the SIMD version of the programs over the theoretical speedup. These specialized instructions are:
- meta-instructions: these instructions perform multiple operations at a time, the multiply accumulation is the most known example. The instruction that computes $\frac{1}{\sqrt{x}}$ is only found in the SIMD ISA and is used in vector normalization in the 3D geometry pipeline.
- *"vector if"* without branch: by using masking and selecting instructions the programmer can perform conditional tests on individual vector elements without any branch instruction.
- Type conversion, vector packing and unpacking. SIMD instruction sets provide float to int and int to float conversions and packing, unpacking instructions with saturated and unsaturated arithmetics. Saturated arithmetics and type conversions are generally high cost operations and are much used in multimedia algorithms that work on floats or 32-bit integers and that need the results to be converted into 8-bit integers to be displayed at screen.

The SIMD instruction sets use large register files with currently four times more space than conventional register files (same amount of registers, 4 times wider). This additional register space can be used in algorithms by maximizing the amount of data stored in registers and then exploit temporal locality at the register level. That reduces the first level cache overall number of miss and may raise significantly the performance of some algorithms.

Table 1. Datasets for the 9 benchmarks, number of elements per AltiVec vector (width) and instruction count reduction (ICR: executed instructions PPC/ AltiVec). Performance of the PPC and prefetch, AltiVec, AltiVec and prefetch versions relative to PPC version and prefetch impact on AltiVec version (PF=prefetch, AV=AltiVec

Benchmark	Dataset Size	Description	Width	ICR	PPC	AV	AV+pf (pf impact)
SOAD	1 MB	1 MPEG2 frame	16	16.3	1.05	2.71	3.44 (1.27)
RGB2YCbCr	9 MB	1 TVHD frame	16	9.8	1.00	4.34	5.95 (1.37)
IDCT	8 MB	8 MPEG2 frame	8	9.4	1.01	5.31	7.12 (1.34)
Img sparse	9 MB	1 TVHD frame	16	5.1	1.17	1.22	1.98 (1.62)
Img dense	9 MB	1 TVHD frame	16	18.4	1.01	2.18	3.63 (1.66)
FIR	10 MB	2400000 elts	4	3.5	1.00	3.02	3.20 (1.06)
DAXPY	8 MB	2000000 elts	4	4.0	1.90	1.12	1.93 (1.72)
Max	16 MB	4000000 elts	4	3.6	1.10	1.63	5.93 (3.65)
3D	10 MB	512000 vertices	4	4.3	1.07	2.18	2.98 (1.37)

5 Performance Evaluation

We have measured the execution time of the 9 micro-benchmarks on a PowerMac G4 450 with 1MByte of L2 cache. We have run each benchmark in its C version and AltiVec version with prefetch on and off. All the results are given as speedups against the C version without prefetch. We have used the datasets presented in Table 1. For all the benchmarks, the L1 misses are only cold misses because multimedia applications exploit mainly spatial locality via data streams. The speedups range from 1.8 to 4.9 for single precision floating-point kernels and from 1.9 to 7.1 for integer ones. The results are shown in table 1. Generally we have measured a speedup lower than the theoretical one. It is between 2.75 and 3.5 for floating point applications. FIR has the biggest instruction count reduction because there is no need for data reorganization because of the intra-iteration parallelization. Its speedup does not reach the theoretical one because the loop cannot be unrolled so the AltiVec functional units stay unoccupied most of the time. The 3D kernel that uses inter-iteration parallelization needs data reorganizations, that is an overhead in instruction count. The dependency graph between instructions is the same in AltiVec than in C (omitting the Permutations for reorganization) so the speedup is essentially limited by the overhead of the reorganization. AltiVec provides an independent permutation unit so a part of the reorganizations can be done in parallel with the computations. Max and DAXPY are more memory intensive than computation, so the speedup is limited by the memory hierarchy performance (cf. next section). For the integer benchmarks, IDCT is very computational and well parallelizable without any overhead, the number of executed instructions is reduced by a 9.4 factor (shown in table 1) and the speedup is 7.1. The sparse filter isn't much efficient because it performs many computations for nothing, just because the data structure is sparse. The AltiVec instruction count is near the one for the dense filter, versus a 4 times reduction for the PPC version. The two last benchmarks have an instruction reduction count of about 16. The 3D kernel uses some AltiVec specialized instructions as $\frac{1}{\sqrt{x}}$ in the normals transformation process. The clipping step uses intensively masking and selection instructions to perform "vector if". The phong lighting step uses vector packing, float to int conversions and saturated arithmetics. The part of the phong algorithm that uses these instructions has a speedup of 14, and the phong lighting step has a global speedup of 7.4. The functions we have implemented in our 3D kernels represent 85% of the time of the geometry transformations into the Mesa 3D graphic Library. The integration of our AltiVec functions into Mesa should give over a 100% increase in performance.

6 Latency Problems: Streaming Prefetch Impact

Our benchmarks are multimedia applications that have streaming data acess. We use blocked algorithm to avoid cache misses other than cold ones. In all our algorithms we have measured the impact of AltiVec streaming prefetch. We

have chosen to launch streaming prefetch each 32Bytes of data. The impact on performance of enabling prefetch is shown in Figure **??**. The impact of prefetch is much more important for AltiVec programs than PPC one but DAXPY.

DAXPY is memory bandwidth limited in both AltiVec and PPC version, and this limit is reached when activating prefetch (the performance are the same for PPC with prefetch than AltiVec with prefetch) so this explains why the speedup due to prefetch is lower for AltiVec version (the version without prefetch is faster in AltiVec).

Max in its PPC version is limited by the branch mispredictions that reduce the average number of instruction fetched each cycle. We see that the impact of prefetch is not very high because memory latency impact is much lower than branch mispredictions impact. Max has no mispredicted branches in its AltiVec version so the fetch does not limit the performance. The limiting factor becomes the memory latency. This explains the great impact of prefetch in the AltiVec version of this Kernel.

There are some kernels on which the activation of the streaming prefetch has a negative impact on the performance. The activation of prefetch generates a heavy load on the Load/store unit and on SOAD and the dense image filter (PPC) it has a negative impact on performance. On the other side, FIR performs many computations per data and it is not much dependent of the memory latency and the impact of prefetch on performance remains low for both versions.

For the PPC version the impact is greater in the sparse version of the image filter than for the dense version. In the dense version the first level cache hit rate is very high without prefetch (1 miss / 32 load), for the sparse version it is only 1/8 so the prefetch increases relatively more the L1 hit rate for the sparse filter. For the AltiVec version the acceleration due to prefetch are very similar because the data are accessed by 128 bit vectors even if they will not be processed (sparse filter). There are a little more computation in the sparse version (masking and selecting the element to be processed) so the impact of prefetch is reduced compared to the dense filter (memory access are less important).

For the rest of the kernels the impact of streaming prefetch for PPC versions is under 10% and between 27% and 66% for AltiVec ones. This is a great improvement in performance for AltiVec programs at a low programmer's effort cost.

7 Memory Bandwidth the Bottleneck of AltiVec Performance

We have seen that streaming prefetch has a great impact on AltiVec programs performance. AltiVec programs are much more sensible to memory latency than the PPC ones, but what impact has memory bandwidth? With streaming prefetch enabled is it the main factor limiting the performance of the AltiVec programs. We couldn't change the available memory bandwidth on our G4 platform, so we used Motorola G4 simulator with the same kernels to evaluate the

impact of memory bandwidth on our micro-kernels. We have used pitsTT6 execution trace as input to the simulator. We have simulated the various memory bandwidth by changing the processor/memory ratio and the SDRAM timings because there was no other ways of modifying the relative memory bandwidth available. We have simulated a halved, doubled and quadrupled memory bandwidth. We have simulated a perfect memory hierarchy too (all data in L1 cache) to get a upper bound of the performance and see how far of this bound we stay.

All the results are in Figure 1. By looking to the speedup due to perfect memory hierarchy we can know if the kernel is memory bound or computation bound. SOAD, RGB2YCbCr and 3D are memory bound in their AltiVec version and DAXPY and Max both in PPC and AltiVec version. On most of the benchmark

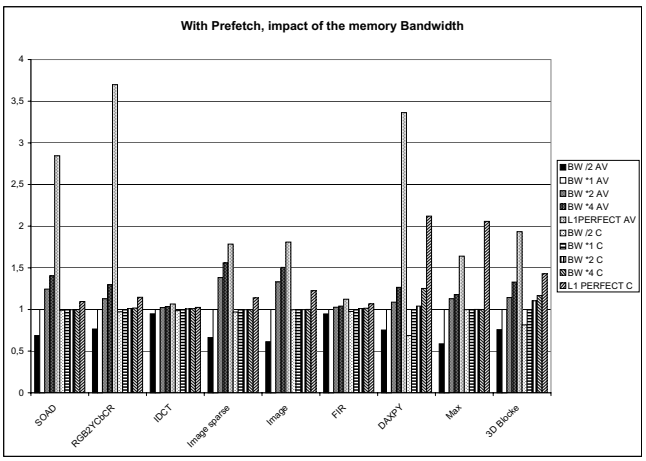

Fig. 1. Impact of the memory bandwidth for the 9 kernels, streaming prefetch on(BW=Memory Bandwidth, AV=AltiVec, L1PERFECT=All data preloaded in first level data cache)

memory bandwidth increase has much more impact on AltiVec performance than PPC. It is not a surprising result because the time spent in computations relatively to the memory accesses in AltiVec programs is much lower than in PPC due to SIMD processing. DAXPY and 3D kernel are sensible to memory bandwidth increase in both C and AltiVec version, but the impact is greater for AltiVec versions. SOAD, RGB2YCbCr, Image filters and Max are sensible to memory bandwidth increase only in their AltiVec versions when prefetch is activated. The two remaining programs IDCT and FIR are essentially computation bound, the impact of memory bandwidth is still greater in AltiVec than in PPC but it is quite negligeable.

The increase of main memory bandwidth has more impact when prefetch is activated for both C and AltiVec versions. When prefetch is used, memory access

are more regular and optimized, the computations are accelerated because of the memory latency reduction, so the need for high bandwidth is more important.

With a quadrupled bandwidth the half of the programs have performance comparable to the performance with perfect memory hierarchy. SOAD, RGB2YCbCr, DAXPY, Max, 3D kernel have more requirements than a quadrupled memory bandwidth.

That shows that the main factor limiting the performance of multimedia, DSP and 3D applications with AltiVec is the main memory bus bandwidth. Currently the PowerPC G4 is limited by its 100MHz 64bit with SDRAM main memory bus on the studied applications. The only tested applications that have really no need for an increased memory bandwidth are applications that are only computation bound. The G4 bus is well designed for most of our applications when we don't use AltiVec but does not meet the requirements for the AltiVec versions. The memory technology that offer a doubled memory bandwidth are currently available with DDRSDRAM and RAMBUS. By using such memory technology some application performance would increase of up to 40% if the latency stays the same as the SDRAM (it is the case for DDRSDRAM).

8 Conclusion

We have optimized 9 multimedia application kernels for AltiVec SIMD instruction set and streaming prefetch. The improvement over original applications range from 1.9 to 7.1 with prefetch enabled on a PPC G4 450. The main factor limiting the performance of the SIMD media kernels is the memory hierarchy. Most applications are computation bound in their original version and become memory bound in their SIMD version. With AltiVec streaming prefetch enabled, the impact of memory latency is reduced and the main limiting factor for SIMD kernels becomes the memory bandwidth. We have shown that the memory bandwidth curently available on PowerPC G4 is sufficient for original kernels, but is the main bottleneck for 7 of our kernels. We show that a quadrupled memory bandwidth that can be achieved with current memory technology like DDRSDRAM or DRDRAM improves the performance of our kernels up to 55%.

References

1. R. Bhargava, L. John, B. Evans, and R. Radhakrishnan. Evaluating mmx technology using dsp and multimedia applications. In *Micro 31*, 1998. 440
2. Brian Case. 3dnow boosts non-intel 3d performance. *Microprocessor Report*, 12(7):18–21, juin 1998. 439
3. Barton Sano Chia-Lin Yang and Alvin R. Lebeck. Exploiting instruction level parallelism in geometry processing for three dimmensional graphics applications. In *Micro 31*, November 1998. 440
4. Jean-Luc Bechennec Julien Sebot and Nathalie Drach-Temam. A performance evaluation of multimedia kernels using altivec streaming simd extensions. *Third Workshop on Computer Architecture Evaluation Using Commercial Workloads (CAECW00) at HPCA-6*, january 2000. 440

5. Motorola. Mpc7400 risc microprocessor hardware sepcification. Technical report, Motorola, sept 1999. 441
6. H. Nguyen and L. K. John. Exploting simd parallelism in dsp and multimedia algorithm using altivec technology. In *ICS99*, 1999. 440
7. P. Ranganathan, S. Adve, and N. Jouppi. Performance of image processing with general purpose microprocessors and media isa extensions. In *ISCA 26*, may 1999. 440