SPEC OpenMP benchmarks on four generations of NEC SX parallel vector systems.

Matthias S. Müller[1]

HLRS, University of Stuttgart, D-70550 Stuttgart, Germany,
`mueller@hlrs.de`,
WWW home page: `http://www.hlrs.de/people/mueller`

Abstract. We describe the performance characteristics of the SPEC OMP benchmarks on parallel vector supercomputers. Points of interest are vectorization, scalability and the comparison between different generations of the same family of NEC SX vector supercomputers. We relate the different performance development of the 11 different applications to different hardware properties of the machine and also to results of the EPCC microbenchmarks. Of special interest is the fact the the NEC SX parallel architecture is not cache consistent.

1 Introduction

The SPEC OMP benchmarks[1] are a set of OpenMP parallel applications that can be used to measure the performance of shared memory systems. Submitted results exist for a variety of vendors. However, since the vendors typically try to submit the best possible result the benchmarks are normally executed with the number of threads that achieve the highest performance on a given system. Therefore only a few scalability results exist. There have been research papers on the performance characteristics on small shared memory machines[3, 2], large SMP systems [5, 6] and on pseudo vector machines like the Hitachi SR8000 [7]. The goal of this paper was to gain insight into the behavior of this set of applications on parallel vector machines like the NEC SX-series.

2 System Description

The NEC SX series is a family of shared memory parallel vector supercomputers. All members of the family have a crossbar that connects the CPUs to the shared memory. An overview of the characteristics can be seen in Tab. 1. In addition, the SX4 has a memory system consisting of SSRAM memory, whereas the SX-5 and SX-6 use DRAM technology. The number of banks in the SX-4 is also significantly higher. To achieve a balanced system with regard to the memory performance the number of CPUs was reduced to 16 and 8 CPUs in the SX-5 and SX-6 respectively. Unlike most SMPs the SX systems are not cache consistent, therefore problems like false-sharing should not exist. The only cache consistency is between the CPU caches and the vector units. The number of memory banks is the same for SX-6 and SX-8. To reach twice the memory bandwidth the bank busy time should be reduced by a factor two. This is not exactly the case, especially for the version with DDR2 RAM that was used here. There is a version with FC-RAM that has better bank busy times. However, there are modifications in the memory system like a bank cache and improved stride 2 memory access that might compensate this effect.

	SX-4	SX-5e	SX-6	SX-6+	SX-8
Peak Performance	2 GF	4 GF	8 GF	9 GF	16 GF
Memory Bandwidh/CPU	16 GB/s	32 GB/s	32 GB/s	36 GB/s	64 GB/s
Clock	125 MHz	250 MHz	500 MHz	563 MHz	1GHz
max Nr. of CPUs	32	16	8	8	8
total Memory Bandwidth	512 GB/s	512 GB/s	256 GB/s	288 GB/s	512 GB/s

Table 1. Basic performance characteristics of the NEC SX family.

3 Short description of the SPEC OMP Benchmark suite

The SPEC OMP benchmark suite consists of 11 applications. With the exception of gafort they have their roots in the SPECfp 2000 benchmark suite. Just a short description of the application is provided here, for a more detailed description see [1]. Wupwise is a quantum chromodynamics code, it consists of 2400 lines of Fortran code. Swim is a program for shallow water modeling. It is a small program with 435 lines of F77 that is known to be memory intensive. Mgrid is a simple Multigrid solver with 489 lines of F77 computing a three dimensional potential field. It was adopted by SPEC from the NAS Parallel Benchmarks. Applu is a parabolic/elliptic PDE solver consisting of 3980 lines of F77. Five coupled nonlinear PDEs are solved on a 3 dimensional, structured grid. An implicit pseudo-time marching schemed is used, based on two-factor approximate factorization of the sparse Jacobian Matrix. This is functionally equivalent to a nonlinear block SSOR iterative scheme with lexicographic ordering. Galgel performs a fluid dynamics analysis of oscillatory instability. Equake calculates a seismic wave propagation. Apsi is a meteorology code, which calculates pollutant distribution. Gafort is a integer intensive genetic algorithm code. Fma3d is a finite element crash code. Art performs an image recognition using a neural network. It is written in C. Ammp is a molecular dynamics code in the field of computational chemistry, it is also implemented in C.

4 Porting to NEC SX systems

The most difficult part in porting the benchmarks was the compilation of the SPEC tools on the target platform. This step was necessary because no binaries are provided by SPEC. The tools are a collection of perl scripts and GNU utilities like make and diff to control the build and run process. One problem in porting the tools was that the NEC systems only have very limited support of shared libraries and dynamic linking, a feature used by the Perl tools.

The compilation of the 11 application codes was almost straightforward. The only required modification was the replacement of the assignment `OMP_LOCK_KIND = selected_int_kind(18)` with `OMP_LOCK_KIND = 8` in `gafort.f90`, because the selected integer kind of the first construct was not supported by the compiler.

It should be noted that our goal was not to reach the maximum performance, but to understand the performance characteristics. Many of the benchmarks have been performed on loaded machines, where only gang-scheduling and resource management ensured that the benchmarks were not affected by other applications running on the machine. Nevertheless, resources like the memory crossbar were shared.

All vectorization was done automatically by the compiler. No directives were inserted and no source code modifications were applied to improve the performance.

Name	Lang.	Vratio	Vlen	Memory (MB)
310.wupwise_m	F	87.34	58.74	1488
312.swim_m	F	99.75	253.48	1584
314.mgrid_m	F	99.14	211.04	480
316.applu_m	F	81.31	34.17	1520
318.galgel	F	92.57	45.41	272
320.equake_m	C	0.06	9.6	464
324.apsi_m	F	76.70	23.02	1648
326.gafort_m	F	40.25	59.60	1680
328.fma3d_m	F	10.29	8.95	1040
330.art_m	C	32.06	242.14	272
332.ammp_m	C	76.67	102.79	176

Table 2. Properties of the SPEC codes on the NEC SX

Tab. 2 shows the reported performance characteristics on the NEC vector-machines. Three codes (swim, mgrid and galgel) show a vectorization ratio of more than 90%. Five codes have a ratio between 40% and 90% (wupwise, applu, apsi, gafort, ammp). The remaining three codes (art,fma3d and equake) show a vectorization below 40%.

The other important issue is the average vector length. Here, swim, mgrid, art and ammp have a vector length above 100. Only equake and fma3d have a very small vector length below 10. The maximum (and best) vector length would be 256.

Looking at the vectorization of the codes it is clear, that only swim, mgrid and maybe galgel will show good performance on the NEC SX. Better results can be achieved with source code modifications, but this was not the target of this work. Instead of the absolute performance, the relative performance between different generations of SX hardware as well as scalability is the main focus. If the balance between different performance properties is constant, the relative performance grow of all applications would be the same. Any difference in performance grow points therefore to different hardware characteristics that dominates the performance for this kind of applications.

5 Performance Measurements

For the scalability tests the number of threads were increased from 1 up to the number of CPUs in the machine. The results can be seen in Fig. 1 and Fig. 2, the results of a run with one thread on the SX-4 has been set to 1. Tab. 3 shows the relative performance between the different SX generations for one and eight threads. The efficiency of the parallel execution is shown in Fig. 4 and Fig. 3.

The following list summarizes the observations for the different applications:

wupwise: the code scales very well up to 32 threads. Due to the good scalability the total node performance is the same for SX-4, SX-5 and SX-6, because all three generations have the same node peak performance.

swim: This code is known to be very memory intensive. The scalability on SX-8 is limited, reaching only a speed-up of 4.5 on 8 threads. This is due to memory degradation on this platform. Tab. 3 shows that the single CPU of SX-8 is 7.02 times faster than a SX-4 CPU, but the 8 thread result is only 4.31 times faster. As explained earlier this is an effect of the relatively long bank busy time of the DDR2 version of SX-8.

mgrid: This code is also known to be quite memory intensive, with a slightly more irregular access pattern than swim, due to the multigrid method. The scalability is good on all platforms. There is an excellent performance improvement going from SX-6+ to SX-8. This is probably due to the improved stride 2 memory access.

applu: The code has only limited scalability and relatively small performance improvements on new platforms.

galgel: For galgel the hardware performance counters show a significant waiting time due to cache misses. This waiting time increases with the number of threads, resulting in poor scalability. For a vector computer cache misses are normally not an issue, but galgel only has a moderate vectorization ratio with a small vector length, thus scalar performance is important.For galgel the efficiency is less for machines with large CPU numbers due to the poor scalability.

equake: This code has limited scalability, especially on machines with fast CPUs, indicating high synchronization overhead.

apsi: The code shows good scalability and efficiency on all platforms.

gafort: This code shows superlinear scaling on SX-6. Again, gafort is one of the codes with small vectorization ration and vector length. Hardware performance counters confirm that this behavior is due to data cache misses.

fma3d: The code shows good scalability and efficiency on all platforms. Only on the SX-4 with up to 32 threads, the efficiency is lower.

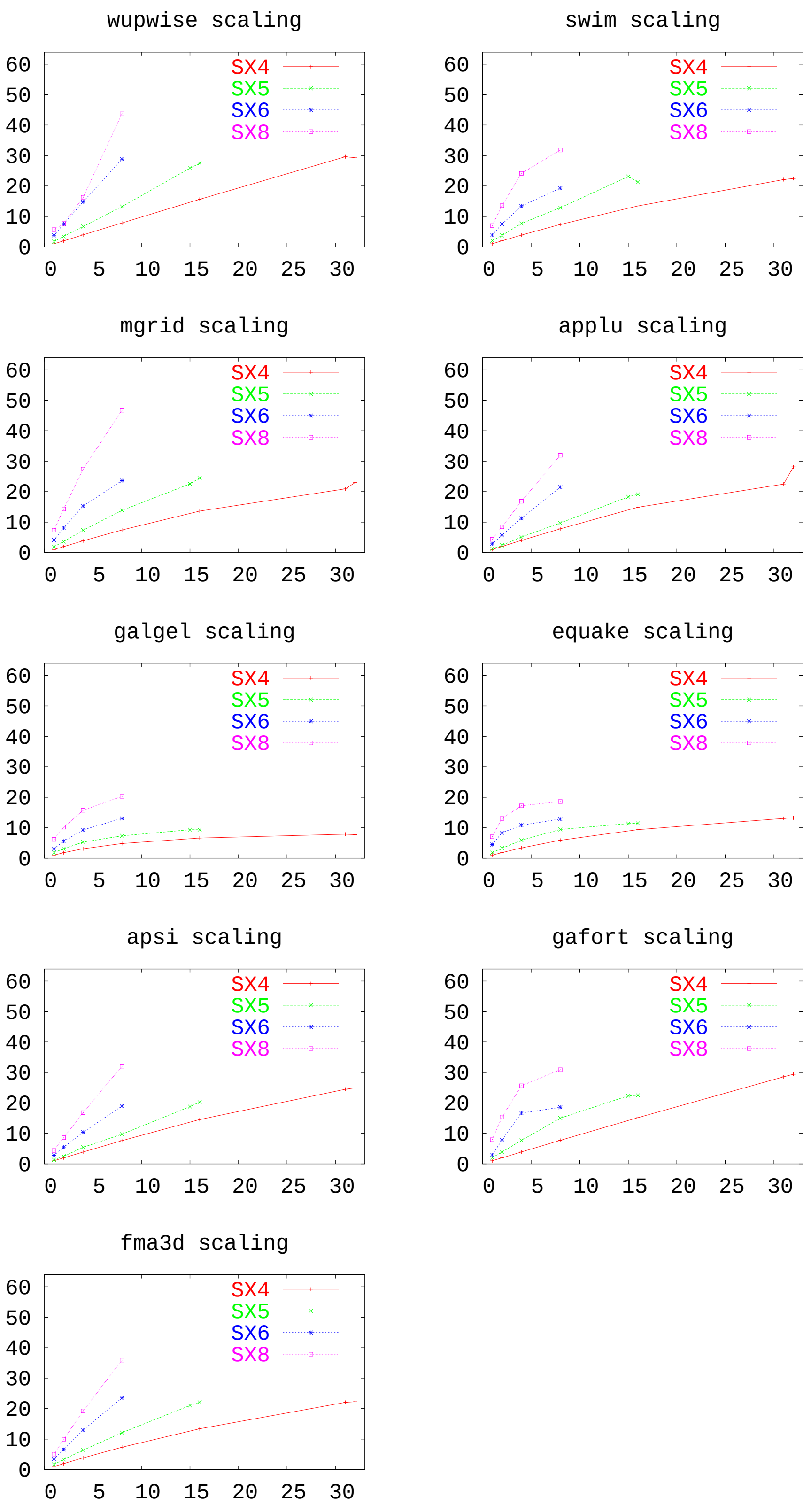

Fig. 1. Scalability of SPEC OMPM applications.

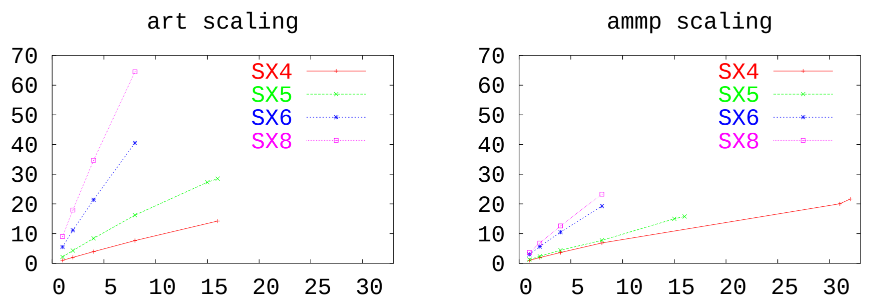

Fig. 2. The application art benefits from new architectural features, whereas ammp shows almost the same performance on SX-6 and SX-8.

art: It was not possible to achieve a successful run with more than 16 threads on the SX-4. Comparing the different generations of SX systems, this code improves better than the peak performance. It clearly benefits from new architectural features introduced by the SX-6.

ammp: This is one of the codes that only show limited improvements between different SX generations: only 3.4 out of 8 when moving from SX-4 to SX-8. One reason is that this code applies a lot of OpenMP locks. On the SX6 the lock overhead is 4.3 microseconds, on SX-8 3.5 microseconds (measured by EPCC microbenchmarks [4]). This ratio perfectly relates to the observed ratio of ammp performance.

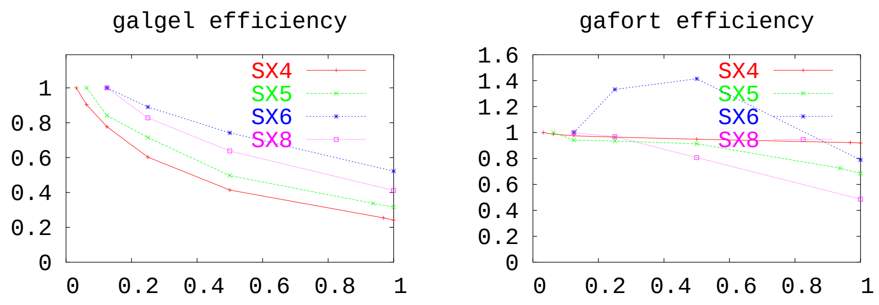

Fig. 3. Efficiency of galgel and gafort depending on the fraction of used CPUs.

6 Summary and Conclusion

With the exceptions of galgel and equake all codes show good scalability. This shows that despite the fact that most current shared memory machines are cache

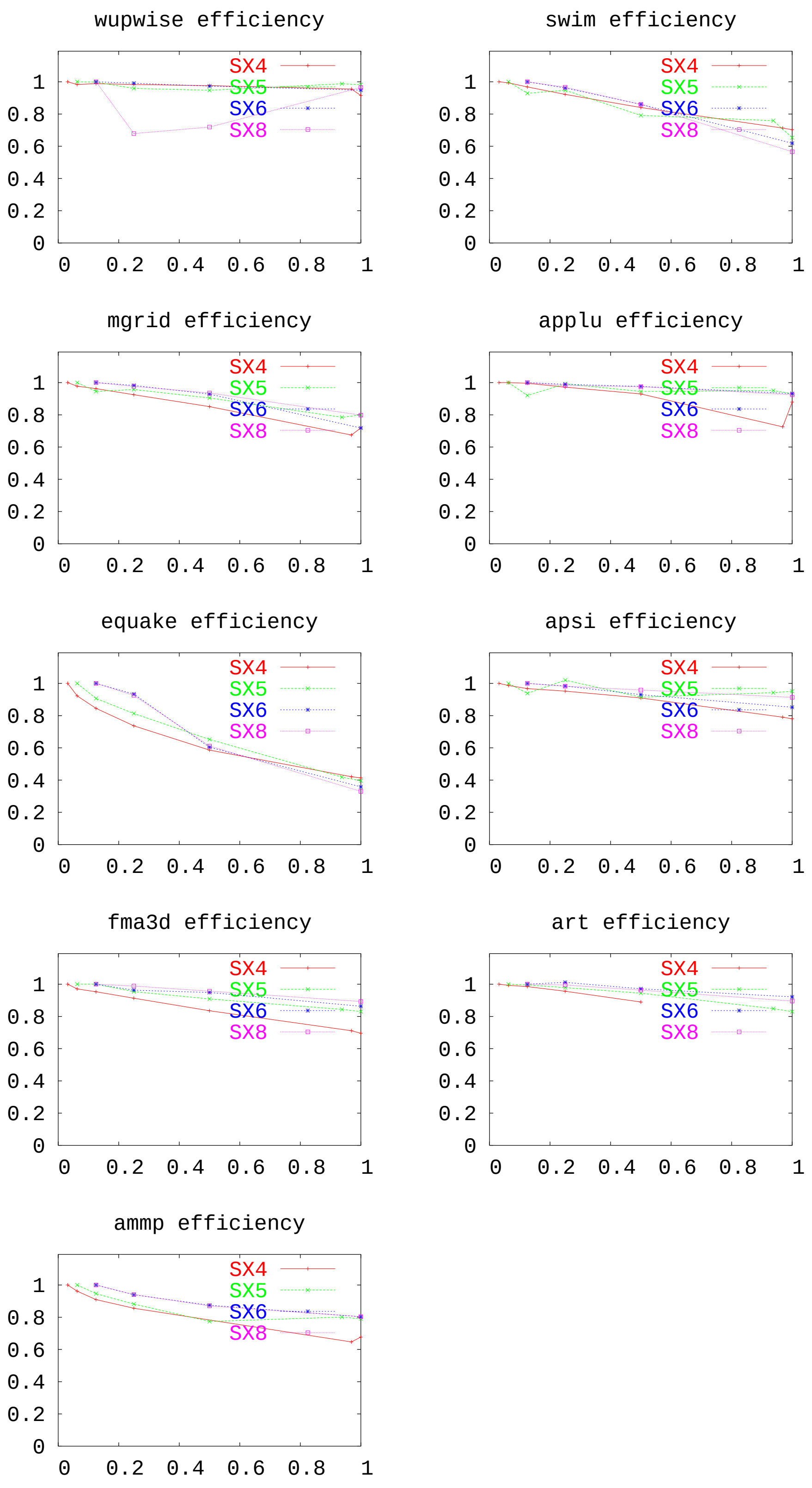

Fig. 4. Efficiency of SPEC OMPM applications.

Name	SX4	SX5	SX6+	SX8
310.wupwise_m	1	1.75	3.80	5.66
312.swim_m	1	2.03	3.89	7.02
314.mgrid_m	1	1.91	4.11	7.32
316.applu_m	1	1.28	2.88	4.32
318.galgel	1	1.85	3.12	6.16
320.equake_m	1	1.81	4.48	7.04
324.apsi_m	1	1.33	2.79	4.39
326.gafort_m	1	2.06	2.94	7.94
328.fma3d_m	1	1.66	3.41	5.03
330.art_m	1	2.15	5.50	9.01
332.ammp_m	1	1.25	3.00	3.62
arith. mean	1	1.73	3.63	6.14

Name	SX4	SX5	SX6+	SX8
310.wupwise_m	1	1.68	3.66	5.56
312.swim_m	1	1.74	2.61	4.31
314.mgrid_m	1	1.87	3.19	6.32
316.applu_m	1	1.25	2.76	4.11
318.galgel	1	1.52	2.70	4.21
320.equake_m	1	1.60	2.17	3.16
324.apsi_m	1	1.27	2.49	4.21
326.gafort_m	1	1.95	2.40	4.01
328.fma3d_m	1	1.65	3.22	4.92
330.art_m	1	2.10	5.30	8.43
332.ammp_m	1	1.12	2.81	3.40
arith. mean	1	1.62	3.03	4.78

Table 3. Relative Performance for 1 thread (left) and 8 threads (right) on different SX models. Obviously different applications benefit to different extent from the faster models.

consistent this is not required to achieve good performance with OpenMP. The single node efficiency is about the same for all generations of SX, showing that the platforms are well balanced.

The absolute application performance gain was also compared to the peak performance improvement. This shows a lack of sustained performance also for vector computers, although they are known to have relatively high sustained performance. However, there is not much difference between vector and scalar codes, showing again that the architectural improvements from one generation to the next are well balanced.

References

1. Vishal Aslot, Max Domeika, Rudolf Eigenmann, Greg Gaertner, Wesley B. Jones, and Bodo Parady. SPEComp: a new benchmark suite for measuring parallel computer performance. In *WOMPAT'01: Workshop on OpenMP Applications and Tools*, volume 2104 of *LNCS*, pages 1–10. Springer, July 2001.
2. Vishal Aslot and Rudolf Eigenmann. Quantitative performance analysis of the SPEC OMP2001 benchmarks. To appear in Scientific Programming.
3. Vishal Aslot and Rudolf Eigenmann. Performance characteristics of the SPEC OMP2001 benchmarks. In *3rd European Workshop on OpenMP, EWOMP'01*, Barcelona, Spain, September 2001.
4. J. M. Bull. Measuring synchronization and scheduling overheads in OpenMP. In *First European Workshop on OpenMP*, 1999.
5. Hidetoshi Iwashita, Eiji Yamanaka, Naoki Sueyasu, Matthijs van Waveren, and Kenichi Miura. The SPEC OMP 2001 benchmark on the Fujitsu PRIMEPOWER system. In *3rd European Workshop on OpenMP, EWOMP'01*, Barcelona, Spain, September 2001.
6. Hideki Saito, Greg Gaertner, Wesley Jones, Rudolf Eigenmann, Hidetoshi Iwashita, Ron Lieberman, Matthijs van Waveren, and Brian Whitney. Large system performance of SPEC OMP2001 benchmarks. In Hans P. Zima, Kazuki Joe, Mutsuhisa

Sata, and Yoshiki Seo adn Massaki Shimasaki, editors, *High Performance Computing, 4th International Symposium, ISHPC 2002*, volume 2327 of *Lecture Notes in Computer Science*, pages 370–379. Springer, 2002.

7. Daisuke Takahashi, Mitsuhisa Sato, and Taisuke Boku. Performance evaluation of the hitachi sr8000 using OpenMP benchmarks. In Hans P. Zima, Kazuki Joe, Mutsuhisa Sata, and Yoshiki Seo adn Massaki Shimasaki, editors, *High Performance Computing, 4th International Symposium, ISHPC 2002*, volume 2327 of *Lecture Notes in Computer Science*, pages 390–400. Springer, 2002.