COMPARING COMPILERS ON INTEL™ PENTIUM 3 and PENTIUM 4 PROCESSORS WITH THE SOM FLOATING POINT ALGORITHM

George Delic *
HiPERiSM Consulting, LLC, Durham, NC
e-mail: **george@hiperism.com**
Web address: **http://www.hiperism.com**
Voice (919) 484-9803 Fax (919) 806-2813

1. INTRODUCTION

This is part of a series of reports on a project to evaluate industry standard fortran 90/95 compilers for IA-32 Linux™ commodity platforms. This report shows results, in a side-by-side comparison for each compiler, for the Intel™ Pentium 3 (P3) and Pentium 4 Xeon (P4) processors for the Stommel Ocean Model.

2.0 CHOICE OF HARDWARE AND OPERATING SYSTEM

Results for the wall clock time are compared for benchmarks compiled using four different Fortran compilers with the Linux™ operating system and one with Windows 2000 (because the Linux™ version was not yet installed). For this project benchmarks were executed in serial mode on a dual processor Intel™ Pentium III (256KB L2 cache) and a dual processor Pentium 4 Xeon 3.06GHz (1MB L3 cache). These architectures offers Streaming Single-Instruction-Multiple-Data Extensions (with version 2, SSE2, for the Xeon). This enables vectorization of loops operating on multiple elements in a data set with a single operation. Where compilers specifically enable SSE/SSE2 it has been tested.

3.0 CHOICE OF COMPILERS

The choice of compilers for Linux™ IA-32 platforms now includes several vendor-supported products. The importance of this category is that vendor products have technical support and undergo continuous development with ports to new architectures as they arrive in the marketplace. The four compilers chosen in this survey are described separately in the following sections and compiler switches used in the two benchmarks are also discussed. However, it is noted here that while all compilers offer a switch to target the Pentium 4, only three (Intel, Lahey, and Portland) offer a specific SSE/SSE2 option (see also notes below).

3.1 Absoft

Absoft f77 and f90/f95 are the Fortran compilers included in the Absoft Pro Fortran™ 8.0 package for Linux™ offered by the Absoft Corporation (http://www.absoft.com). The f90/f95 version has a Cray front-end and resulted from a five-year collaboration with Cray Research. With this compiler use of the –O3 compiler switch enables automatic architecture detection and selection of the Pentium 3 or 4 instruction set.

3.2 Intel

The Intel Fortran Compiler version 8.0 targets both Intel IA-32 and IA-64 (Itanium) architectures, but only the former has been used in this project so far. Details on the compiler features are available at HiPERiSM Consulting, LLC's URL. Code for target architectures generated with either the –tpp6 (Pentium 3) or –tpp7 (Pentium 4) switch.

3.3 Lahey

The Lahey/Fujitsu Fortran 95 compiler (hereafter Lahey) for Linux™ is available from Lahey Computer Systems, Inc., (http://www.lahey.com). The Express version 5.6 for Microsoft Windows 2000™ was used on the P3 and P4 because it was available from another project for the same hardware. With this compiler use of the –tpp compiler switch enables automatic architecture detection for the P3 only. However, release v7.1 (for Windows) and v6.2 (for Linux) support compiler switches –tp4 and –sse2 to target the Pentium 4 Xeon and the SSE2 instruction set. The v6.2 release and the new

switches were studied for this report, but surprisingly, showed timing results that differed only by a few percent from those for the 5.6 version shown here. The use of the SSE2 switch likewise produced negligible difference. This suggests that the 5.6 version of this compiler already performs excellent optimizations of the regular double loop structure on this benchmark where the main arithmetic work is performed.

3.4 Portland

The pgf90™ fortran compiler (Linux™ distribution) from the Portland Group, (http://www.pgroup.com) was used in the CDK 4.0 release where it supports OpenMP, MPI and OpenMP+MPI parallel applications on HiPERiSM's IA-32 Linux™ cluster. With this compiler use of the –fast compiler switch enables automatic architecture detection. Note that the CDK 5.1 release (not used here) may offer additional performance enhancement of the Pentium 4 Xeon processor with the use of SSE2 options.

3.5 Portability and migration issues

Portability issues come up when legacy Fortran code needs to be compiled. In this respect a compiler that allows extensions to the f90/f95 standard can save time and effort. The two compilers that offer the widest scope in portability are those from Absoft and Portland. Compilers from Lahey and Intel are less forgiving of such extensions.

Here we also mention some migration issues that came up with compiler and architecture changes. The change in architecture from P3 to P4 Xeon also involves changes in library versions. As a result, two of the compilers had to either be upgraded or have patches applied. Installation of the Absoft 8.0 compiler for the Xeon processor and the newer Linux Kernel does require download and application of two patch files to resolve glibc version issues (these patch files are available from the Absoft URL given in Section 3.1). Likewise, an attempt was made to install the 7.1 release of the Intel Fortran compiler on the P4 Xeon. However, again version skew with glibc suggested the simpler option of installing the 8.0 release. Whenever the version of a compiler is changed performance is also expected to change. This is especially true of the Intel compiler since major performance improvements are announced with the 8.0 release. Therefore, the changes in performance reported here for the Intel compiler are due to improvements in the compiler technology as well as the change in architecture.

4.0 CHOICE OF BENCHMARKS

4.1 Introduction

The algorithms used here have been executed on a wide variety of platforms and are excellent benchmarks in studying how a compiler and architecture interact for the types of operation they use. A fuller discussion of the benchmarks is available in previous reports (HCTR-2001-1). What follows is only a brief introduction.

4.2 Stommel Ocean Model Algorithm

The Stommel Ocean Model (SOM) is a legacy Fortran 77 code with a compute kernel consisting of a double-nested loop that performs a Jacobi iteration sweep over a two-dimensional finite difference grid. The number of iterations is fixed at 100 and, because the data set is regular and the loop structure is conventional, this code should present compilers with good prospects for vectorization. Therefore, as a floating point algorithm, the SOM is useful in studying performance scaling with problem size N over the N x N grid. Six cases were used in this analysis for data sets with grid dimensions in the range N=2000 (1000) 7000, corresponding to problem size scaling from $N^2=4x10^6$ to $49x10^6$ data points.

5.0 COMPARING EXECUTION TIMES

The following sections summarize execution time with four compilers for the SOM algorithms with six data sets (Cases 1 to 6).

5.1 Timing performance

Whole code execution was measured with calls to the Fortran 90/95 system_clock routine for all compilers as this was deemed to be the most portable and accurate timing method.

5.3 Stommel Ocean Model results

For the SOM algorithm the choice of compiler switches is summarized in Table 5.1. For the Pentium III case the only differences are the use of the p6 target architecture switch and the Intel 7.1 release. Timing results (without SSE enabled) are shown in Tables 5.2 (Pentium 3) and 5.3 (Pentium 4). Figures 1 and 2, for Pentium 3 and

Pentium 4 respectively, show these times as bar charts.

Table 5.1 Compiler command and switches for the SOM algorithm on the P3 and P4 processors.

Compiler and version	Compiler command and selected switches	Effect of switches
Absoft 8.0	f90 –s –cpu:p6 –O3 –ffixed f90 –s –cpu:p7 –O3 –ffixed	Optimize for P3 or P4 Xeon target
Intel 7.1 (P3) Intel 8.0 (P4)	ifc –O3 –tpp6 –FI ifc –O3 –xK –tpp6 –FI ifort –fast –tpp7 –FI ifort –fast –xW –tpp7 –FI	Optimize for P3 target. Vectorize and enable SSE. Optimize for P4 Xeon target. Vectorize and enable SSE2.
Lahey 5.6	lf95 –tpp -fix	Optimize for P3 (also used on P4)
Portland 4.0 (P3 and P4)	pgf90 –fast –Mvect pgf90 –fast –Mvect=sse	Vectorize Enable SSE

Table 5.2 Execution times (seconds) for the SOM algorithm with four compilers on the Pentium III (933 MHz) without SSE enabled.

N	Absoft	Intel	Lahey	Portland
2000	50.0	38.8	36.4	41.4
3000	110.5	94.4	87.7	92.7
4000	197.7	159.6	150.3	163.3
5000	305.3	224.3	246.8	253.1
6000	443.4	320.0	332.0	388.5
7000	586.5	427.6	477.9	524.4

Table 5.3 Execution times (seconds) for the SOM algorithm with four compilers on the Pentium 4 Xeon (3.06 MHz, 1MB L3 cache) without SSE enabled.

N	Absoft	Intel	Lahey	Portland
2000	8.17	5.44	6.77	7.62
3000	20.2	12.7	15.8	17.7
4000	37.5	24.5	30.6	33.0
5000	52.5	37.8	40.1	50.8
6000	72.8	50.1	62.4	71.5
7000	101.6	72.4	85.2	95.4

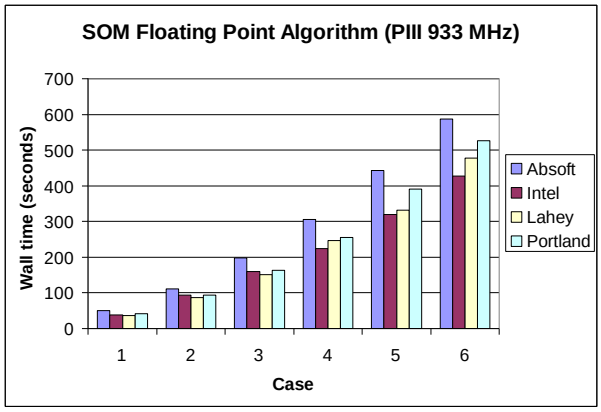

Fig. 1 Execution times of four different compilers for the SOM floating point algorithm (without SSE) on the Pentium 3.

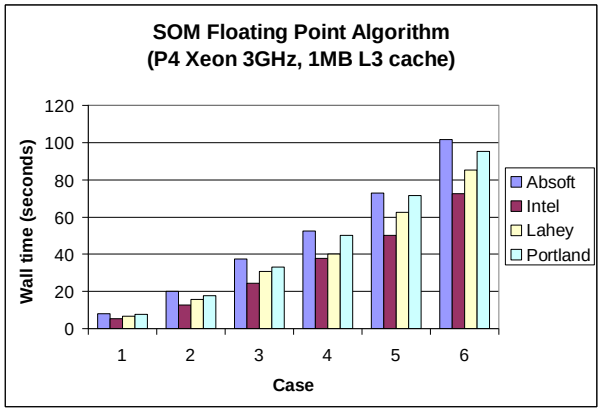

Fig. 2 Execution times of four different compilers for the SOM floating point algorithm (without SSE) on the Pentium 4 Xeon.

It is interesting to observe the changes in performance between the Pentium 3 and 4. On the Pentium 3 the Lahey compiler reported the lowest wall clock times for the first three cases and the Intel compiler does this for the last three cases. On the Pentium 4 the Lahey compiler either equals, or outperforms, the other compilers. To compare the performance gain for each compiler due to a change in architecture Figure 3 shows the ratio of the Pentium 3 execution times to those of the Pentium 4. For all cases the performance gain is at least of the order of a factor of 5, with some variability. It is surmised that this is due to differences in use of processor and cache in each compiler. A deeper performance analysis of the underlying reasons for this is the subject of a future report.

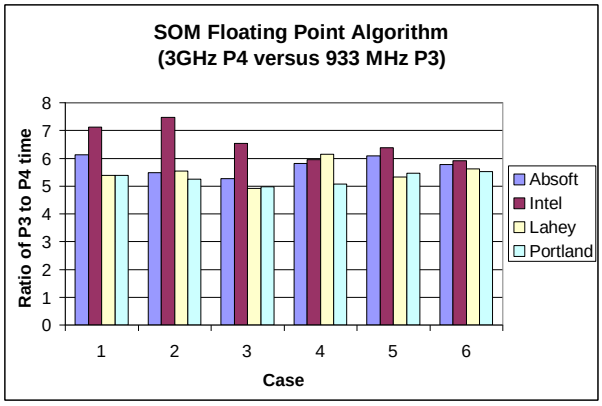

Fig. 3 Ratio of execution times of four different compilers on the Pentium 3 versus the Pentium 4 Xeon for the SOM floating point algorithm (without SSE).

To study overall performance features Figures 4 and 5 for the Pentium 3 and Pentium 4 Xeon, respectively, show the trends in descriptive statistics for the six problem sizes. As expected the mean and standard deviation of the execution time rise for both processors. However, the coefficient of variation (standard deviation divided by the mean) of execution times within this group of compilers initially fluctuates slowly as problem size increases. But for the largest problem size the difference in execution time for different compilers shows a diminishing trend. What is more, the variability in compiler performance is, in general, very much lower for the Xeon processors. For example, in Case 6 the coefficient of variation shows values of 0.135 and 0.0788 for Pentium 3 and 4 respectively. In other words, for problems with regular data structures and workloads dominated by a single loop nest, the differences between compiler performance diminishes as problem size increases, and is even smaller on processors with larger cache sizes.

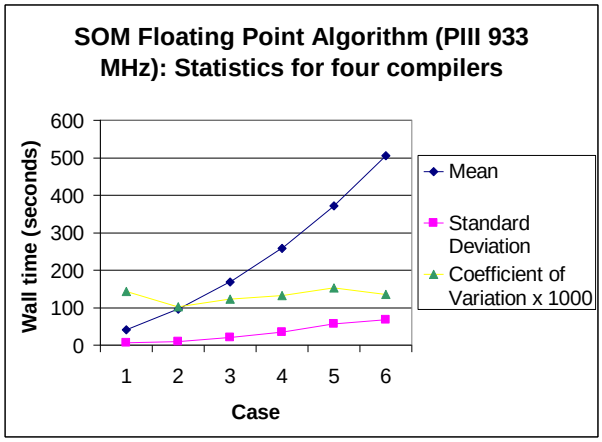

Fig. 4 Execution time statistics for four different compilers with the SOM floating point algorithm on the Pentium 3.

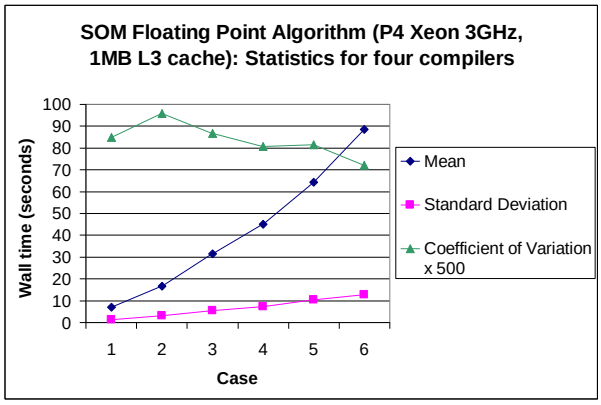

Fig. 5 Execution time statistics for four different compilers with the SOM floating point algorithm on the Pentium 4 Xeon.

6.0 EVALUATION OF SSE RESULTS

Two of the compilers (Intel and Portland) include specific switches to enable the SSE feature of the Pentium III architecture. For regular data structure and vectorizable loops this should produce enhanced performance on this generation of processors.

The SSE options are enabled as indicated in Table 5.1 for the SOM floating point algorithm (note that the SSE option is irrelevant for the Kallman integer and logical algorithm). Figures 6 and 7 for Pentium 3 and 4, respectively, summarize the effect of the SSE for the Intel and Portland compilers. The results of Figure 7 for the Xeon processor show that performance enhancements from SSE are negligible for the Intel compiler and marginal for the Portland compiler. This is in remarkable contrast to the situation on the Pentium 3 shown in Figure 6 where, for regular data structures, and vectorizable code with long loops, dramatic performance enhancements are possible from enabling SSE.

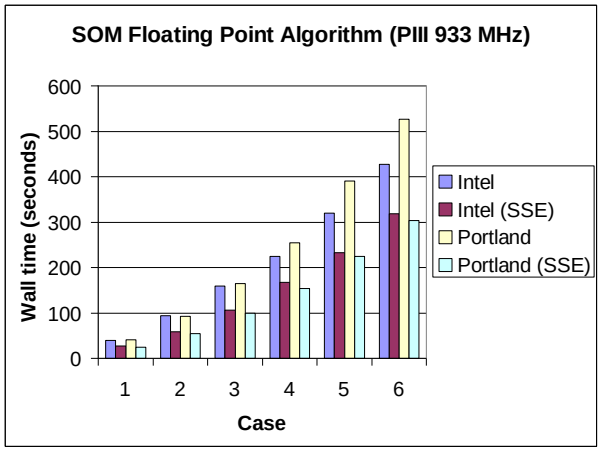

Fig. 6 Execution times of two compilers for the SOM floating point algorithm without and with SSE enabled on the Pentium 3.

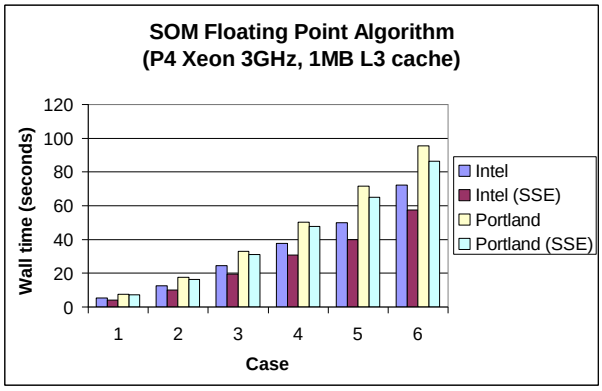

Fig. 7 Execution times of two compilers for the SOM floating point algorithm without and with SSE enabled on the Pentium 4 Xeon.

It is instructive to compare theses results with those for the Pentium II in the first of these reports (HCTR-1999-1) for the Absoft compiler on the Pentium II processors. The Xeon results reported here take some 7 times less execution time. This improvement is due to performance developments in both compiler and architecture technologies.

7.0 CONCLUSIONS

This report presented performance results of four fortran compilers in the IA-32 environment. The variability in performance found was specific to the benchmarks selected and represented extremes in arithmetic operation types. Variability in performance results is expected but the details will depend on the balance of integer, logical, and floating point operations. Real-world codes have different mixtures of such operations. Therefore subsequent reports will study examples of "real-world" code as used in weather, climate, and air quality models.

The analysis in subsequent reports will include in-depth evaluation of performance of this group of compilers with specialized software such as the Intel VTune™ Performance Analyzer. Also in this evaluation the consequences of compiler switches for numerical precision and stability will be investigated.