AERMOD-HPC A HIGH PERFORMANCE COMPUTING VERSION OF AERMOD

George Delic *
HiPERiSM Consulting, LLC, Durham, NC, USA

Arnold R. Srackangast
AS1MET Services, Blanco TX, USA

1. INTRODUCTION

HiPERiSM **C**onsulting, LLC, (Durham, North Carolina) has **l**inked with **AS1**MET Services (Blanco, Texas) to form a joint venture, **HiCLAS1** (http://www.hiclas1.com), dedicated to bringing High Performance Computing (HPC) capability to Environmental Modeling. The HiCLAS1 mission is to develop (or enhance) software and improve performance on current and future computers for legacy Air Quality Models (AQM). The first model chosen for performance enhancement by HiCLAS1 is the U.S. EPA's AERMOD developed by the U.S. EPA Office of Air Quality Planning and Standards (OAQPS), Emissions Monitoring and Analysis Division (EMAD), at the U.S. EPA in Research Triangle Park, North Carolina, U.S.A (SCRAM). In-house Quality assurance testing and results from Beta testers show performance of the serial version of AERMOD-HPC that is 1.95 to 3.43 times faster than the EPA distribution of AERMOD. The purpose of this presentation is to provide quantitative evidence of the measured hardware performance metrics to demonstrate how the improvements in efficiency are achieved. Results with the serial version of AERMOD-HPC are presented for Intel Pentium Xeon processors. The subject of numerical differences is taken up in technical reports available on-line (HiCLAS1).

2. CHOICE OF HARDWARE AND OPERATING SYSTEM

The hardware used for the results reported here is the Intel Pentium 4 Xeon processor with separate platforms using the Linux™ operating systems for both 32-bit and 64-bit platforms, respectively. The hardware used for the results reported here is the Intel Pentium 4 Xeon (P4) and Pentium Xeon 64EMT (P4e) processors.

* *Corresponding author address:* George Delic, HiPERiSM Consulting, LLC, P.O. Box 569, Chapel Hill, NC 27514-0569. Email: george@hiclas1.com

The operating system (OS) is HiPERiSM Consulting, LLC's modification of the Linux™ 2.6.9 kernel to include a patch that enables access to hardware performance counters. This modification allows the use of the Performance Application Programming Interface performance event library (PAPI, 2005) to collect hardware performance counter values as the code executes. Results for selected performance metrics are presented with a view to giving insight into how the application is mapped to the architectural resources by an un-named compiler.

3. BENCHMARK TIMINGS

Four benchmarks are used with the number of sources varying from 10 to 963, number of receptors from 771 to 916, and the number of meteorological hours from 2160 to 8760. Details on the benchmarks are available elsewhere (HiCLAS1). Two version of the U.S. EPA's AERMOD model are used here: the executable distribution, designated AERMOD-EPA, and the version compiled from the (unmodified) source distribution designated as AERMOD-EPA/SRC. The U.S. EPA source and executable are available on-line (SCRAM).

To create the High Performance Computing (HPC) version of AERMOD the source code for the U.S. EPA distribution was progressively modified to enhance performance. Speedup of the HPC version over the EPA model is shown in Fig. 1 (AERMOD-HPCS versus AERMOD-EPA) and Fig. 2 (AERMOD-HPCS versus AERMOD-EPA/SRC). The results of Fig. 2 are for both versions compiled from source with identical compiler options.

Whether comparing against the U.S. EPA executable or source code compiled with the same compiler options AERMOD-HPCS always delivers superior performance. The remainder of this report gives some in reasons as to why this is the case.

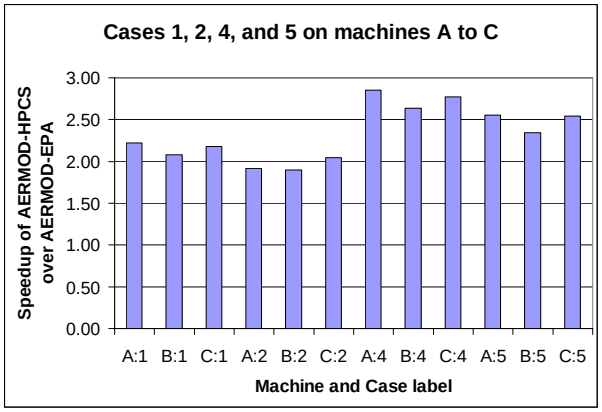

Fig. 1 displays the ratio of runtimes for AERMOD-HPCS and AERMOD-EPA for three Pentium 4 Xeon machines (A to C) with a 32-bit Windows OS. This shows that, for four Cases, performance enhancement ranges from 1.9 to 2.77 times faster than AERMOD-EPA (depending on the platform and data set used in the benchmark).

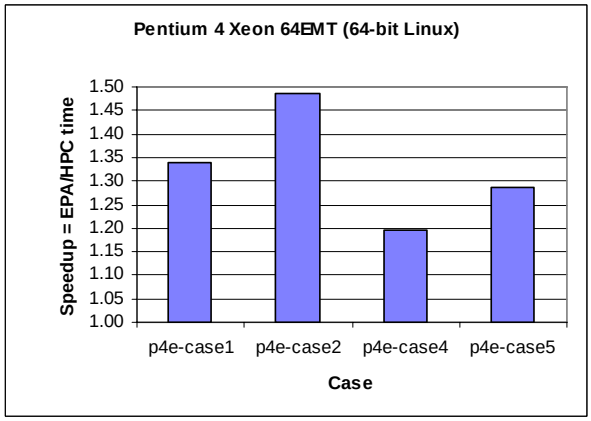

Fig.2. Speedup of AERMOD-HPCS as measured by the ratio of the wall clock time for the U.S. EPA AERMOD version (compiled from source code) divided by the wall clock time for AERMOD-HPCS for the same compiler options for four cases.

4. HARDWARE PERFORMANCE EVENTS

The hardware used for the results reported here is the Intel Pentium 4 Xeon (P4) and Pentium Xeon 64EMT (P4e) processors. For this hardware performance counters were used to measure performance metrics and some values are summarized below.

4.1 Operations and instructions

AERMOD-HPCS delivers higher Mflops rates as is shown in Fig. 3 for the 64-bit Linux case.

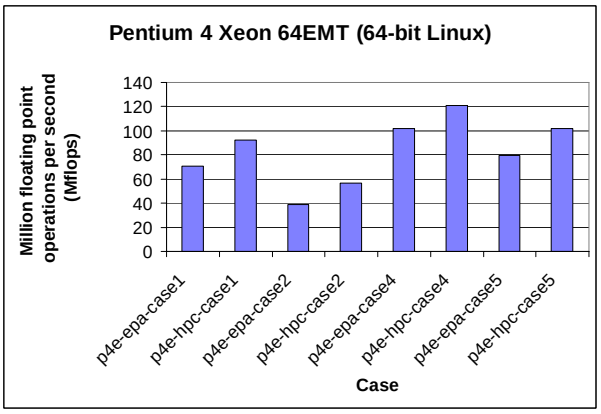

Fig. 3 Mflops for AERMOD in EPA (epa) and HPCS (hpc) versions for four cases.

One important contributing factor to higher Mflops is that AERMOD-HPCS delivers higher vector/SSE instruction rates as shown in Fig. 4. However, performance gains for AERMOD from enhanced vector instructions alone is limited because of the lack of vector loop structure and the predominance of control transfer instructions. These stall the vector pipeline and cycles are lost to loading of new instructions.

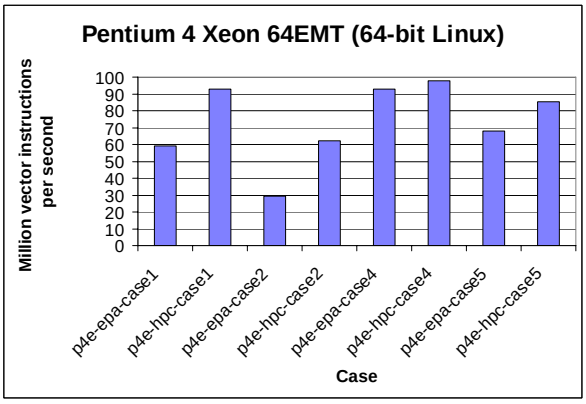

Fig. 4 Vector instruction rates for AERMOD in EPA (epa) and HPCS (hpc) versions for four cases.

4.2 Memory footprint

For AERMOD in general, the rate of total memory instructions issued is voluminous. The consequence of AERMOD's memory footprint is that the path to memory becomes a critical performance bottle-neck. This bottle-neck is somewhat ameliorated in AERMOD-HPCS compared to AERMOD-EPA as is described in the following.

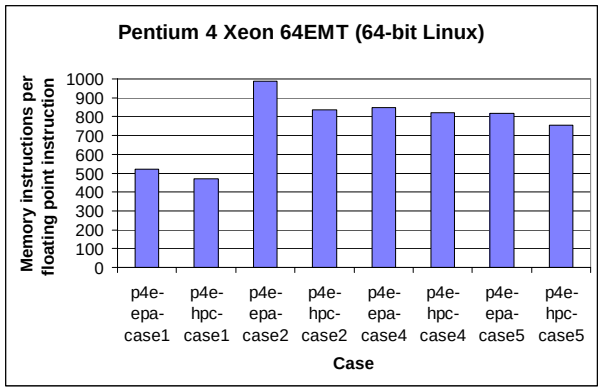

Fig. 5 Memory instructions per floating point instruction for AERMOD-HPCS (hpc) compared to that for the U.S. EPA's distribution (epa) for four cases.

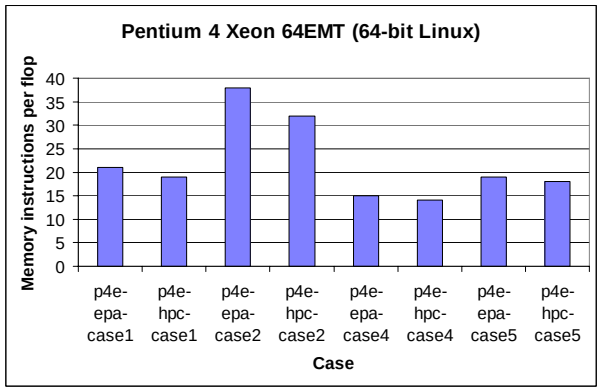

Fig. 6 Memory instructions per flop for AERMOD-HPCS (hpc) compared to that for the U.S. EPA's distribution (epa) for four cases.

Fig. 5 shows the load balance of memory versus floating point instructions and demonstrates the extent to which AERMOD is a memory-bound application. As a consequence, AERMOD is extremely sensitive to any inefficiency in memory access. It is notable that AERMOD-HPCS reduced this load imbalance somewhat, but it is still critical. Fig. 6 shows that for each flop there are more then 14 memory instructions in all cases on either P4 (not shown) or P4e platforms. This is a gross imbalance suggesting that the CPU is starved of data and spends excessive cycles in an idle state.

4.3 Branching instructions

Control transfer instructions are a significant source of lost CPU cycles in AERMOD and chief among these are branch instructions. Mispredicted branch instructions on deep pipelined processors are an important cause of lost performance, because instructions in the mispredicted path are cancelled and operations are not completed. The pipeline is flushed and new instructions are loaded with the result that cycles are lost to arithmetic performance. Fig. 7 shows that in all cases, on both 32-bit (not shown) and 64-bit platforms, AERMOD-HPCS has reduced mispredicted branch instruction rates and this correlates positively with higher Mflops.

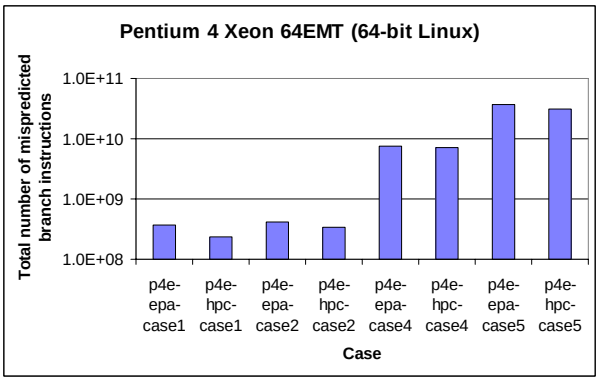

Fig. 7 Logarithm of total number of mispredicted branch instructions for AERMOD-HPCS (hpc) compared to that for the U.S. EPA's distribution (epa) for four cases.

4.4 TLB Cache usage

The translation lookaside buffer (TLB) is a small buffer (or cache) to which the processor presents a virtual memory address and looks up a table for a translation to a physical memory address. If the address is found in the TLB table then there is a hit (no translation is computed) and the processor continues. The TLB buffer is usually small, and efficiency depends on hit rates as high as 98%. If the translation is not found (a TLB miss) then several cycles are lost while the physical address is translated. Therefore TLB misses degrade performance. In the case of AERMOD it is the instruction TLB misses that are critical. Higher instruction TLB miss rates suggest that the processor pipeline stalls more frequently because of a higher rate of control transfer instructions. This is due to numerous procedure calls and voluminous mispredicted branch instruction rates.

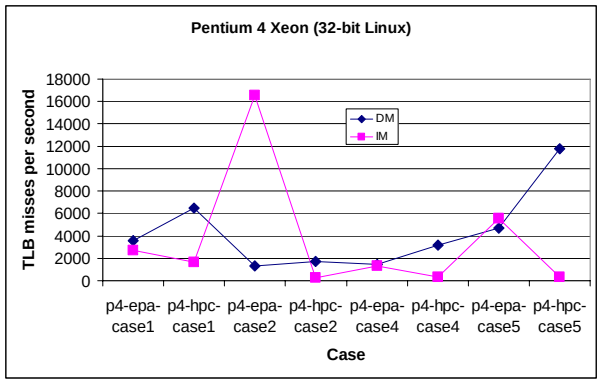

Fig. 8 TLB misses per second for AERMOD-HPCS (hpc) compared to that for the U.S. EPA's distribution (epa). Note that the instruction misses (IM) are reduced in AERMOD-HPCS for each of four cases.

While the TLB data miss rates (DM) have increased in AERMOD-HPCS relative to the EPA version, performance has improved, suggesting that it is the TLB instruction miss rates that are important for performance in AERMOD. The AERMOD-HPCS version is more efficient in reducing instruction TLB miss rates (IM) through optimization and resource allocation compared to the EPA version. The most dramatic reduction is in Case 2 for the 32-bit platform, as shown in Fig. 8, and this explains (in part) why AERMOD-HPCS has so much better performance compared to the EPA version (see Case 2 in Fig. 2).

4.5 L1 Cache usage

A cache miss occurs when data or instructions are not found in the cache and an excursion to higher level cache, or memory, is necessitated. Cache misses result in lost performance because of increasing latency in the memory hierarchy. Memory latency is smallest at the register level and increases by an order of magnitude for a L1 cache reference, and another order of magnitude to access L2 cache. In the case of AERMOD this analysis will focus on the L1 cache behavior. Fig. 9 shows L1 cache miss rates for data (DCM) and instructions (ICM). Even though the ICM rate has been scaled the reduction for AERMOD-HPCS versus the EPA version is evident and has a positive correlation with the TLB instruction miss rate reduction shown in Fig. 8.

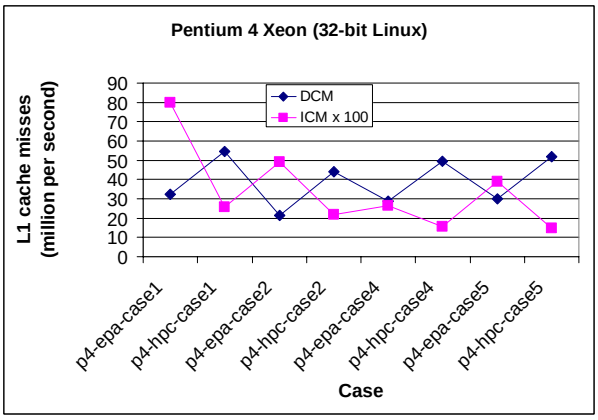

Fig. 9 Million L1 cache misses per second for AERMOD-HPCS (hpc) compared to that for the U.S. EPA's distribution (epa) for four cases.

5. WHY IS AERMOD-HPC FASTER?

The code transformation applied in AERMOD-HPCS take cognizance of procedures occurring at the leaves of a deep calling tree. Such procedures invariably have no loop structure but consist of simple arithmetic statements and conditional code blocks. The most frequently called procedures typically have little arithmetic work. These are some of the reasons for lack of vectorizable loops and the high rates of branching instructions in the U.S. EPA version of AERMOD. As a result the extremely high instruction TLB misses for AERMOD are a critical source of performance limitations. High memory instruction rates are due to high TLB instruction miss rates and also to correlated L1 instruction cache miss rates. This behavior is ameliorated by the improved efficiency of the AERMOD-HPCS version in reducing the performance consequences of this behavior. AERMOD-HPCS is faster than the U.S. EPA version AERMOD-EPA/SRC because it delivers:

- Higher Mflops rates
- Lower number of memory instructions per floating point instruction
- Lower mispredicted branch instruction rates
- Lower instruction TLB miss rates
- Lower L1 instruction cache miss rates

6. CONCLUSIONS

This performance analysis of the U.S. EPA version of AERMOD, shows that it is a memory intensive application with large rates of control transfer instructions such as branching logic and procedure calling overhead. These features result in large observed rates for branching instructions and instruction TLB misses. These, in turn, result in stalled pipelines and cycles lost to arithmetic operations. In combination these characteristics of the AERMOD code place a limit on the optimal performance possible from it on commodity platforms. This is because, by design, commodity hardware solutions offer a cost effective compromise between processor clock rates, cache size, and bandwidth (or latency) to memory.

The AERMOD-HPCS version goes some way to ameliorate these performance limitations. As a result gains in computational efficiency translate into reduced wall clock time. However, there is still scope for further improvements and progress will be reported in subsequent reports at the HiCLAS1 URL where the AERMOD-HPCS version of AERMOD is available at the download pages (HiCLAS1).

References

HiCLAS1: Further details are available from the HiCLAS1 URL at http://www.hiclas1.com.

PAPI, 2005: *Performance Application Programming Interface*, http://icl.cs.utk.edu/papi. Note that the use of PAPI requires a Linux kernel patch (as described in the distribution).

SCRAM: AERMOD is available at U.S. EPA, Technology Transfer Network, Support Center for Regulatory Air Models http://www.epa.gov/scram001/.