2.1

PORTING AIR QUALITY MODELS TO NEXT GENERATION COMMODITY PLATFORMS

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

1.0 INTRODUCTION

This report to the Air Quality Modeling (AQM) community has two goals (a) review the consequences to Air Quality Modeling of the revolution in commodity hardware technology, and (b) evaluate the latest releases of fortran 90/95 compilers Linux™ on commodity platforms with AERMOD [1] and CAMx [2] benchmarks.

2.0 HARDWARE REVOLUTION

The next revolution in commodity hardware has arrived and those who learn to ride the wave of this revolution will become the performance leaders on commodity hardware. Virtually all the major hardware vendors now offer CPU's with dual core technology and it is anticipated that by the end of 2006 quad core CPUs will be announced. By 2010 it is expected that there will be more than 100 cores per CPU. The consequences for AQMs will be an order of magnitude loss in performance when executed on such processors if they remain the predominantly scalar performers that they are at this time. These developments place pressure on software developers in AQMs to seek conversion to appropriate parallel computing models. Furthermore, clusters now represent one half of the technical computing market, with a rapid decline in shipment of 32-bit platforms and a corresponding wider acceptance of 64-bit commodity hardware. Also, some 73% of high-end computing platforms use Linux as the OS of choice. These revolutionary changes motivate HiPERiSM Consulting, LLC's initiative to (a) measure hardware performance of AQMs, (b) initiate work on serial-to-parallel conversion of serial AQMs, and (c) explore the consequences of new software support of parallel computing.

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

3.0 SOFTWARE EVOLUTION

Three compilers commonly in use for AQMs have each had a major release within the last year. These are Absoft (10.0), Intel (9.1), and The Portland Group, STMicroelectronics (6.1). HiPERiSM Consulting, LLC's evaluation of these new releases with AQMs showed some interesting progress in performance and therefore it was decided to include some new results and compare them with those presented in previous years.

4.0 BENCHMARKS

The hardware used for the results reported here is the Intel Pentium 4 Xeon (P4) and Pentium Xeon 64EMT (P4e) processors. These have processor clock rates of 3GHz and 3.4GHz, respectively. Each is in a dual configuration with a corresponding front side bus (FSB) of 533MHz and 800HMz shared by each pair of processors. 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. However, in this report no performance data (other than runtime) will be discussed, as results have been presented elsewhere (cf. Delic in [3,4]). To conserve space the choice of compiler switches is not listed here and full details on them may be found at the technical reports pages of the HiPERiSM Consulting, LLC, URL [5].

The choice of benchmarks includes AERMOD 04300 and CAMx 4.03. For the latter we use a prior release because results with the earlier compiler releases were available for comparison against the present (newer) compiler releases. Furthermore, it has been our experience with later

releases of CAMx that performance of the compute kernels continues to be similar. A benchmark with CMAQ [6] awaits the availability of release 4.6 in Q4CY006.

4.1 AERMOD 04300

AERMOD version 04300 was provided by the U.S. EPA. Two data sets are used in this benchmark. The first (EPA-E2) was provided by the U.S. EPA, and the second (ENV-T1) was provided by an environmental consulting company. Fig. 1 shows the comparison of three compilers on the P4 and P4e platforms for the ENV-T1 case. Fig. 2 shows a comparison of the previous and current release of each compiler on the P4e platform for the EPA-E2 case. With the exception of the Absoft compiler the new release of each compiler does show an improved performance. The current performance leader for AERMOD is the Intel v9.1 compiler on both platforms.

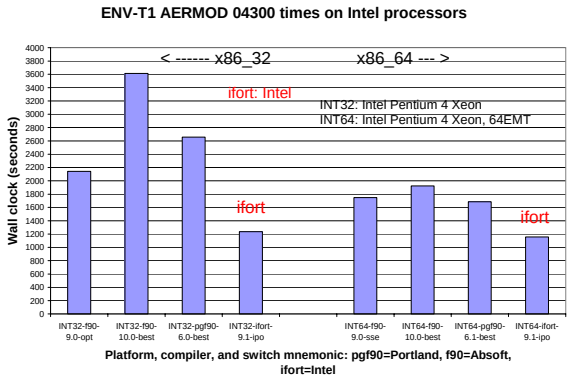

Fig. 1 Run time for AERMOD benchmark ENV-T1 with three compilers on two platforms. The 32 bit (P4) platform results are shown on the left and those for the 64 bit platform (P4e) on the right. The switch mnemonics are defined in a Technical Report (in HCTR2006_1 at [5]) where the corresponding compiler switches are listed. The best time reported in this group is for the Intel v9.1 compiler on the P4e platform.

4.2 CAMx

The CAMx code developed by ENVIRON (http://www.camx.com) is a Fortran 77 code for an Eulerian photochemical model that is widely used in the AQM community. The benchmark results reported here were for one day (08/22) of the 2000 episode in the Greater Metro area with the base5a.regular.GOES base case for 2000 using the reported non-equilibrium emissions and the GEOS-satellite correct meteorology. Modeling files

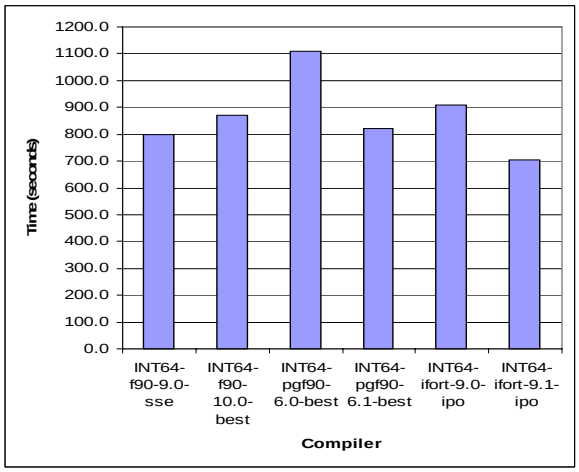

Fig. 2 Run time for AERMOD benchmark EPA-E2 with three compilers on the P4e platform. This figure compares performance of the compilers with results of the latest releases and previous release, in each case. The switch mnemonics are defined in a Technical Report (in HCTR2006_1 at [5]) where the corresponding compiler switches are listed. The best time reported in this group is for the Intel v9.1 compiler.

are obtainable from the TCEQ site at http://www.tnrcc.state.tx.us/air/aqp/airquality_photomod.html#section4> http://www.tnrcc.state.tx.us/air/aqp/airquality_photomod.html#camx>

Fig. 3 shows runtime results with some missing values. Some observations are in order for the missing cases. The Absoft v10.0 compiler benchmark is in progress at this time and will appear at a later date. The Intel ifort v9.1 result for the P4e platform is missing because the compiler fails with an “Internal error” warning. A bug report has been posted with the Intel support site. An update of the results for Fig. 3 will be posted in a Technical Report (in HCTR2006_2 at [5]).

4.3 Analysis of Benchmark Results

The performance analysis of these benchmarks, shows several important features of current compiler technology for commodity hardware. For AERMOD and CAMx benchmarks

- performance results with three compilers show as much as 77% variability (depending on the platform).
- very significant changes in performance occur between even minor releases of a specific compiler: by as much as 22%-26% (with some exceptions).
- Performance on 64-bit platforms is superior to that on 32-bit platforms: ranging from 7% (Intel) to 87% (Absoft) improvement in runtime (for AERMOD, and less so for CAMx).

It is clear that compilers for commodity platforms have undergone a rapid maturation process in less than two years and continue to exchange leadership in performance as new releases arrive.

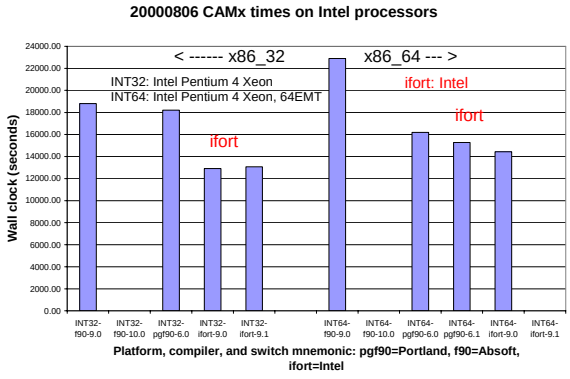

Fig. 3 Run time for CAMX benchmark 20000806 with three compilers on two platforms. The 32 bit (P4) platform results are shown on the left and those for the 64 bit platform (P4e) on the right. The compiler switches used are defined in a Technical Report (in HCTR2006_2 at [5]). The best time reported in this group is for the Intel v9.0 compiler on the P4 platform.

5.0 TRACKING THE REVOLUTION

In the following sections the revolution in commodity computing is reviewed.

5.1 Trends in hardware

Two important recent developments in commodity hardware have important consequences for software development with commodity clusters. The first of these is the proliferation (and rapid acceptance) of processor hardware that supports 64-bit memory addressing. Examples include processors such as the Intel Pentium Xeon 64EMT [7], American Micro Devices (AMD) Opteron [8], and IBM's Power PC G5 [9]. Each of these architectures supports 64-bit Linux kernels, and also Apple Mac OSX 10.4 (in the case of the G5). In this discussion we say little about the Itanium processor because we know of only two customers who use it. The second recent development is the availability of multi-core processors. At this time (2006) these are dual core CPU's with separate GPR's, functional units and cache hardware. Dual core processors are already in the market place and examples of third party vendors offering such solutions with AMD dual core processors are Microway [10], HPC Systems [11] and SUN Microsystems [12]. It is anticipated that developments in multi-core processors will be rapid in the next few years and by 2010 the number of cores per processor is expected to exceed 100 (cf. Jack Dongarra in [4]).

Intel [7] has announced a technology roadmap of processor fabrication with feature resolution ranging down from 90nm (in 2003) to 22nm (in 2011). This represents an increase in density from the current (2005) to a future (2011) of approximately (65/22)^2 = 8.7, or nearly an order of magnitude. Intel quad core CPU's will become available in 2007. Furthermore, for the first time since the late 1990's Intel has designed two new motherboards to target technical computing.

5.2 Trends in software

As in the past, the two dominant parallel programming paradigms are MPI [13] and OpenMP [14] so our emphasis will be on these and products that support them. The arrival of multi-core CPUs is coincident with the availability of large memory capacity so we anticipate a resurgence of interest in software development for large memory models within a Shared Memory Parallel (SMP) programming paradigm such as OpenMP. The OpenMP application program interface (API) is now at the 2.5 standard [14] and discussions are underway on what features should be included in the 3.0 standard. The OpenMP model is supported by all major compilers and has been endorsed by key applications developers. It has a bright future in riding the wave of the multi-core revolution especially since OpenMP debug tools such as Intel's Threadchecker™ [7] are available to developers in helping them detect memory leaks.

5.3 Performance and code structure

Effective and efficient parallel processing depends on a combination of several key factors:

- Efficient serial performance
- Achieving good vector/parallel scalability
- Macroperformance, or gross behavior of the computer-application combination
- Microperformance, or the underlying factors responsible for the observed macroperformance
- Usability, or program development environment.

The above is what we said seven years ago and the story has not changed. What makes the search for effective parallel processing on commodity processor clusters particularly challenging is how (and with what frequency) technology changes.

Before parallel performance (Macroperformance) is evaluated the serial performance (Microperformance) of an application needs to be optimized. By serial performance we mean the on-core execution efficiency of the code. For example, code that does not take advantage on the extended SSE instruction set, or otherwise experiences serial performance bottle-necks, should not be executed in parallel mode before serial performance is optimized. Achievable serial execution efficiency will depend on several factors that change with time: memory architecture, FSB rates, CPU architecture (e.g. number of stages in the pipeline), scope of hardware resources (e.g. number of GPR's, size of the TLB cache, etc), and the instruction set that comes with each new hardware generation. Performance of applications can be precisely measured from hardware counters available on commodity processors at the user level.

Defining good versus poor performance (cf. Delic in [15]) depends on the criteria applied. The basic performance categories that need to be examined are:

- floating point operations
- integer and logical operations
- memory operations
- I/O operations

Specific metrics need to be defined and examined in each category. HiPERiSM uses multiple metrics that show either rates (i.e. number per unit time), or ratios (ratio of operations or instructions of different categories).

For optimal performance code must be structured to present compilers with ample opportunity to engage SSE instructions and overlap with memory (or I/O) operations. This means applying the usual practices for vector code construction:

- Introducing DO loop code blocks that compilers can easily identify as potentially vectorizable
- Removing any I/O from potential vector code blocks
- Removing any vector inhibitors from potential vector code blocks
- Simplifying the calling tree within potential vector code blocks
- Eliminating or simplifying conditional segments within potential vector code blocks

There is nothing new in these basic practices for constructing vector code - they are the same that applied when serial code was ported to Cray vector architectures. They are also the appropriate rules for constructing code with good parallel potential, e.g. applying task parallelism to the DO loop code block.

5.4 Problems on commodity solutions

The fundamental problem with commodity hardware solutions is the performance cost of accessing memory (cf. Delic in [3,4]). Processor performance has improved by leaps-and-bounds in the last decade or so, whereas memory latency has not improved on a corresponding scale [16]. As a consequence, the challenge of optimizing the balance between memory operations and arithmetic operations is crucial because commodity architectures compromise on memory bandwidth and latency to reduce costs.

Applications with a voluminous rate of total memory instructions need to be examined carefully. A high rate of memory instruction issue need not be an indicator of a performance bottleneck. Benchmarks with good vector character that deliver of the order of 1Gflop on a

Pentium 4 Xeon can also show high memory access rates. But if an application has low vector instruction rates and voluminous memory access rates (as do AERMOD and CAMx), performance is constricted on commodity architectures where memory bandwidth is limited by the FSB and cache design.

5.5 Riding the wave of the revolution

With the advent of multi-core CPUs, for models that already use MPI, exploration of hybrid MPI and SMP levels of parallelism could be beneficial for performance scaling. Hybrid parallel computing models have been explored in the past, but their utility with dual processor commodity environments has been limited. Within the next year we will see the possibility of SMP models with 8 (or more) threads per node applied to real world models. This means that in a hybrid parallel model there will be a need to measure parallel scalability as a function of the number of MPI processes versus the number of OpenMP threads. For AQMs the performance consequences of multi-core hardware architectures are not known at this time and will need to be studied in detail. Such studies will be necessary on each new platform because (a) the memory latency and interconnect fabric will be different in each case, and (b) different compilers will give different performance results.

6.0 CONCLUSIONS

This performance analysis of AERMOD and CAMx shows that compilers must be tested with each new release because performance differences are very significant (even between minor releases). Furthermore, it can no longer be assumed that one single compiler will provide superior performance indefinitely. In fact, even within the space of a year, dominance in performance can change between different compilers. However, one note of warning needs to be sounded: even though a new release of a compiler can give dramatically improved performance in one application is may not do so in another. These rapid developments in compiler releases mean also that comparisons between compilers can show a "leap-frog" effect as new releases arrive asynchronously from vendors. Thus, active users of commodity compiler products are well advised to exercise new compiler releases as they arrive with their favorite benchmarks. Furthermore, since new (major) compiler releases often come with compiler bugs, the best advice to end-users is to apply at least two different compilers to the same benchmark in mission critical projects to validate numerical accuracy.

Developments in hardware and software offer the opportunity of orders of magnitude increases in performance of AQMs. However, they also require refined programming practices from cluster users coupled with precise measurement of performance metrics using hardware performance counters. Model developers should plan for transition of existing models to multi-core parallel architectures and evaluate the software options with respect to suitability to this task based on the criteria of level-of-effort, usability, and scalability. Such a plan should have as focal points:

- ***Level-of-effort:*** A assessment of the typical cost of parallelizing serial production code.
- ***Usability:*** The base of legacy serial code requires simpler parallelization strategies that do not require a complete rewrite of the code as the first step.
- ***Scability:*** For multi-core architectures message passing programming may either be avoided entirely by use of OpenMP, or in combination with it, in a hybrid mode.

Prototyping with easy-to-use parallelizing compilers and OpenMP, provides input to a decision making process on the serial-to-parallel conversion strategy. HiPERiSM's experience with OpenMP shows that the level of effort is some five to ten times less than that in using MPI. Furthermore the use of tools available for OpenMP programmers may further reduce the level of effort. For large applications hybrid parallel models that use OpenMP and MPI in combination will determine the potential for scalability with clustered SMP nodes as the number of cores per CPU scales upward.

Citations

[1] AERMOD is available at U.S. EPA, Technology Transfer Network, Support Center for Regulatory Air Models http://www.epa.gov/scram001/.
[2] CAMx was developed by ENVIRON Corp. and is available at http://www.camx.com.
[3] 6th International Conference on Linux Clusters: The HPC Revolution 2005, Chapel Hill, NC, April 26-28, 2005,

http://www.linuxclustersinstitute.org/Linux-HPC-Revolution/Archive/2005techpapers.html.
[4] 7th International Conference on Linux Clusters: The HPC Revolution 2006, Norman, OK, May 2-4, 2006.http://www.linuxclustersinstitute.org/Linux-HPC-Revolution/Archive/2006techpapers.html
[5] HiPERiSM Consulting, LLC, http://www.hiperism.com.
[6] CMAQ was developed in the Atmospheric Modeling Division (AMD) of the NOAA Air Resources Laboratory (ARL) in collaboration with the U.S. EPA's National Exposure Research Laboratory (NERL) and is distributed by CMAS at http://www.cmascenter.org.
[7] Intel Corporation http://www.intel.com
[8] American Micro Devices http://www.amd.com
[9] IBM http://www.ibm.com
[10] Microway http://www.microway.com/opteron.html
[11] High Performance Computing Systems http://www.hpcsystems.com/servers.htm
[12] SUN Microsystems http://www.sun.com
[13] The Message Passing Interface (MPI) standard, http://www.mcs.anl.gov/mpi/index.html.
[14]OpenMP http://www.openmp.org
[15] Commodity Cluster Symposium, Baltimore, MD, July 26-27, http://www.arl.hpc.mil/events/Clusters2006
[16] STREAM http://www.cs.virginia.edu/stream