CMAQ PERFORMANCE WITH THREE COMPILERS ON TWO PLATFORMS

George Delic*
HiPERiSM Consulting, LLC, P.O. Box 569, Chapel Hill, NC 27514, USA

1. INTRODUCTION

This presentation continues a decade-long study of CMAQ behavior when compiled and executed with vendor-supported compilers on commodity hardware platforms. In the past CMAQ has been ported to compilers from the Portland Group® [PGI] and (more recently) the Intel Corporation® [INTEL]. We propose that the time has come to add other compilers to the CMAQ fold: specifically new results are presented for Absoft Fortran from the Absoft Corporation® [ABSOFT]. A great deal has changed over ten years in compiler development, and while compilers from different vendors tend to leap-frog each other in performance, all have undergone ground-breaking evolution in following hardware developments. These developments are not reflected in the limited choice of compilers and the options available in the standard releases from the CMAS Center download site [CMAS].

This report presents results of comparing standard U.S. EPA and the multithreaded version of CMAQ developed by HiPERiSM Consulting, LLC, for multi-core processors with three compilers on two platforms (Delic, 2009, 2010). The Rosenbrock (ROS3) chemistry solver version of CMAQ 4.7.1 is chosen for this analysis because it offers the best potential for parallel performance.

The ROS3-HC code is a hybrid parallel model with three levels of parallelism. The (outer) Message Passing Interface (MPI) level is the one previously delivered in the standard U.S. EPA distribution. The (inner) parallel layers developed at HiPERiSM have added both thread-level parallelism and instruction-level parallelism (at the vector loop level). This report examines parallelism in CMAQ at the MPI and the thread levels.

2. CHOICE OF PLATFORMS

2.1 Hardware

The hardware systems chosen were the platforms at HiPERiSM Consulting, LLC, shown in Table 2.1. Each of the two platforms, Intel and Advanced Micro Devices (AMD), have a total of 8 and 48 cores, respectively. This cluster is used for either MPI only, or hybrid thread-parallel OpenMP plus MPI execution, and results for both modes are reported below.

Table 2.1. Platforms at HiPERiSM Consulting, LLC

Platform	AMD	Intel
Processor	AMD™ Opteron 6176SE	Intel™ IA32 W5590
Peak Gflops (SP/DP)	110.4 / 55.2	106.6 / 53.3
Power consumption	105 Watts	130 Watts
Cores per processor	12	4
Power consumption per core	8.75 Watts	32.5 Watts
Processor count	4	2
Total core count	48	8
Clock	2.3GHz	3.33GHz
Band-width	42.7 GB/sec	64.0 GB/sec
Bus speed	1333 MHz	1333 MHz
L1 cache	64KB	64KB
L2 cache	512 KB(1)	256MB
L3 cache(2)	12MB	8MB
(1) Per core, (2) Per socket		

2.2 Compilers

This report concurrently evaluates the latest compiler versions from Intel (12.0), Portland (11.5) and Absoft (11.1) for CMAQ 4.7.1 on 64-bit Linux operating systems using top-of-the-line performance hardware from Intel and AMD. The U.S. EPA and ROS3-HC multi-threaded parallel version were compiled and executed with all three compilers on both platforms shown in Table 2.1.

For each compiler several groups of optimization switches were test, but only a few are selected for presentation here. These correspond to two groups for Absoft (abs3 and abs4), and one respectively for Intel (ifc) and Portland (pgf) compilers. For each compiler group this analysis included new builds of CMAQ support libraries such as NetCDF, IOAPI, MPICH, STENEX and PARIO. In each case compiler options have been chosen after extensive research into three areas of fundamental significance for CMAQ: the memory model, numerical precision, and time to completion. This study found that although the highest optimization levels produce the shorter runtimes, in some cases they also introduce numerical differences that compromise numerical precision for a small (10%) subset of the species concentration value population. This observation

* *Corresponding author:* George Delic, george@hiperism.com.

leads to a choice of more conservative compiler option groups. Details of compiler options are available from the author.

3. EPISODE STUDIED

For all CMAQ 4.7.1 results reported here the model episode selected was for August 09, 2006, using data provided by the U.S. EPA. This episode has the CB05 mechanism with Chlorine extensions and the Aero 4 version for PM modeling. The episode was run for a full 24 hour scenario on a 279 X 240 Eastern US domain at 12 Km grid spacing and 34 vertical layers.

4. RESULTS FOR U.S. EPA's CMAQ 4.7.1

4.1 Serial runtime results

Fig 4.1 shows the serial (one MPI process) runtime results for U.S. EPA's CMAQ 4.7.1 release with the three compilers introduced in Section 2.2. The numerical values are shown in Table 5.1. The values for the AMD platform shown in Fig 4.1 have been reduced by a half to better fit the figure scale. The AMD Absoft compiler group abs4 is not yet completed.

On the Intel platform, the Intel compiler delivers, the shortest runtime (25.1 hours) followed closely by the Absoft compiler (25.9 hours) with group abs4. On the AMD platform Absoft (52.3 hours) outperforms both Intel and Portland compilers even with the lower optimization choice (abs3). The difference between Intel and AMD platforms (e.g. 27.1 versus 52.3 hours for group abs3) is mainly due to the differences in clock speed, bandwidth, and cache size (Table 2.1).

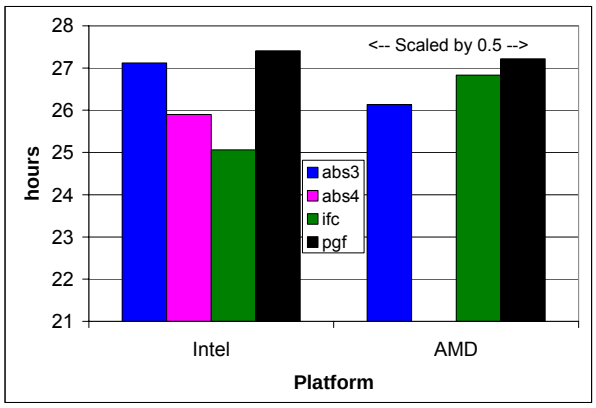

Fig 4.1: Serial runtime (in hours) for CMAQ 4.7.1 in the U.S. EPA release of the ROS3 solver for the Intel and AMD platforms. The AMD results have been reduced by a factor of 2 to better fit the scale of the figure.

4.2 MPI Performance

The remaining results discussed in this section are for the Portland compiler on Intel and AMD platforms. MPI results for Intel and Absoft compilers are in progress and will be reported at a later date. For fastest MPI runtimes all tests are performed locally (i.e. on-node) to avoid any interconnect latency issues, using 8 to 48 MPI processes. The Intel platform is limited to 8 cores whereas the AMD platform has up to 48 cores.

Typical runtime results for the standard U.S. EPA distribution of CMAQ 4.7.1 are shown in Table 4.1 for Intel and AMD platforms with the Portland™ compiler.

Table 4.1. Wall clock times (in hours), MPI parallel speedup, and MPI efficiency, for the CMAQ Rosenbrock solver. This is for the U.S. EPA's standard release of CMAQ 4.7.1 on HiPERiSM's Intel and AMD platforms for the Portland compiler group pgf.

Col x Row = NP	Time in hours (EPA)		MPI speedup versus NP=1		MPI parallel efficiency	
	Intel	AMD	Intel	AMD	Intel	AMD
1 x 1 = 1	27.4	54.4	1.00	1.00	1.00	1.00
1 x 2 = 2	15.6	29.2	1.76	1.86	0.88	0.93
2 x 2 = 4	8.5	15.7	3.22	3.47	0.81	0.87
2 x 4 = 8	5.1	8.5	5.39	6.41	0.67	0.80
4 x 4 = 16		4.41		12.3		0.77
4 x 8 = 32		2.49		21.9		0.68

Results on both platforms show that CMAQ gains from the evolution of commodity computer architectures. It is interesting to note that the AMD platform with 8 cores equals the performance of the Intel platform with 4 cores and exceeds it with 8 (Intel) versus 16 (AMD) cores.

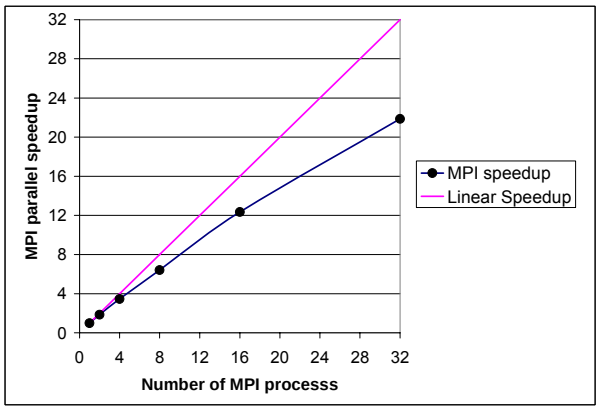

Fig 4.2: MPI parallel speedup with the Portland compiler for CMAQ4.7.1 in the U.S. EPA release of the ROS3 solver for the AMD platform.

Using the results of Table 4.1, Fig. 4.2 shows the MPI parallel speedup. This displays an

increasing divergence from linearity with increasing number of MPI processes and is best analyzed by inspecting MPI parallel efficiency.

4.3 MPI Efficiency

This section presents MPI parallel efficiency results for the Rosenbrock (ROS3) chemistry solver in the U.S. EPA version of CMAQ 4.7.1. The MPI parallel efficiency shown in Table 4.1 is the MPI speedup divided by the number of MPI processes. Fig. 4.3 summarizes CMAQ 4.7.1 MPI parallel efficiency (and inefficiency) with increasing process count. From the logarithmic scale on the abscissa it is clear that the ROS3 solver shows an exponential decline in MPI parallel efficiency when NP>1. Using the extrapolation shown in Fig. 4.3 the asymptote of parallel efficiency is of the order of 60% for 64 MPI processes where CPUs are idle for 40% of the wall clock time (on average).

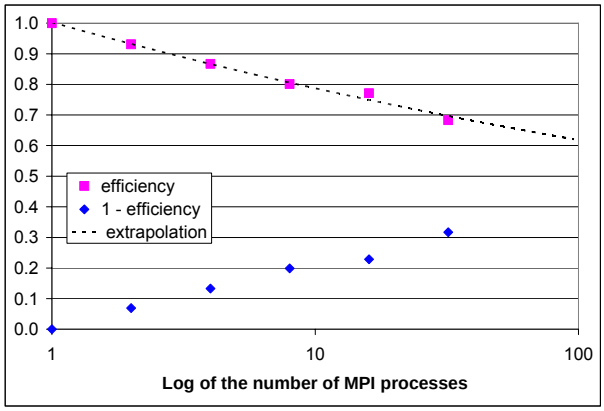

Fig 4.3: MPI parallel efficiency (and inefficiency) for CMAQ4.7.1 in the U.S. EPA release of the ROS3 solver for the AMD platform. Note the logarithmic scale on the abscissa.

Efficiency and inefficiency may be translated to the average utilization and idle times and these are shown in Fig. 4.4 as a percentage of the total runtime at each MPI process count. While decrease in runtime for CMAQ 4.7.1 is dramatic with increasing MPI process count, the corresponding increase in idle time is relentless. The trend line in Fig. 4.3 suggests a decline to 40% MPI parallel efficiency in the next decade of the abscissa's logarithmic scale (100-1000). Using 80% as an acceptable efficiency threshold, Table 4.1 suggests a maximum allowable MPI process count on each platform of 4 (Intel) and 8-16 (AMD), respectively.

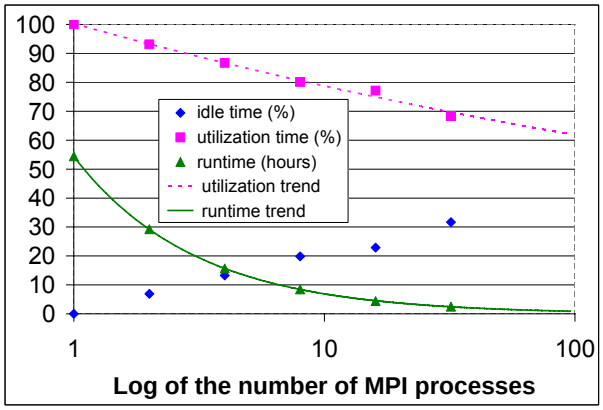

Fig 4.4: MPI parallel runtime, percent utilization and idle time, for CMAQ4.7.1 in the U.S. EPA release of the ROS3 solver for the AMD platform. Two trend lines are included. The ordinate represents both hours for runtime and percent for utilization and idle time fractions. Note the logarithmic scale on the abscissa.

5. HYBRID OpenMP+MPI RESULTS

5.1 CMAQ runtime

Results for runtime in the hybrid MPI+OpenMP version of CMAQ 4.71.1 with three compilers, are presented here for the parameter choices BLKSIZE=1536 and NCMAX=48. For a discussion of these parameters see the previous reports in this series (Delic, 2009,2010). Table 5.1 summarizes the CMAQ 4.7.1 results for runtime (in hours) for the case of one MPI process.

Table 5.1. Wall clock times (in hours) for the U.S. EPA (ROS3-EPA) and hybrid MPI+OpenMP (ROS3-HC) versions of CMAQ 4.7. The platforms are Intel and AMD for the Absoft (abs3, abs4), Intel (ifc) and Portland (pgf) compilers.

Compiler group	Platform	ROS3-EPA	ROS3-HC				
			Time in hours by thread count				
			1	4	8	12	16
abs3	Intel	27.1	36.1	21.8	19.3		
abs4		25.9	35.8	21.1	18.7		
ifc		25.1	29.4	21.3	19.2		
pgf		27.4	31.1	23.3	21.9		
abs3	AMD	52.3	67.5	43.1	38.3	37.2	35.4
ifc		53.7	63.2	45.5	40.7	40.2	39.0
pgf		54.4	60.9	44.4	43.0		

In this table execution times of the standard U.S. EPA release are in the column labeled ROS3-EPA. Columns under the label ROS3-HC show results of the hybrid MPI+OpenMP CMAQ 4.7.1 version with the Rosenbrock solver. The rows correspond to the different compiler choices

for an MPI process count of 1 and the thread count is the number appearing under the column labeled as ROS3-HC. The blank cells indicate that results are not yet available at this time, or are limited by 8 cores per node on the Intel platform.

For the CMAQ hybrid parallel version ROS3-HC Figs 5.1 and 5.2 respectively show the Intel and AMD platform results of Table 5.1 for 1, 4, and 8 cores. The best times with 8 threads are those for the Absoft compiler, and, although gains are less when more threads are added, this lead increases.

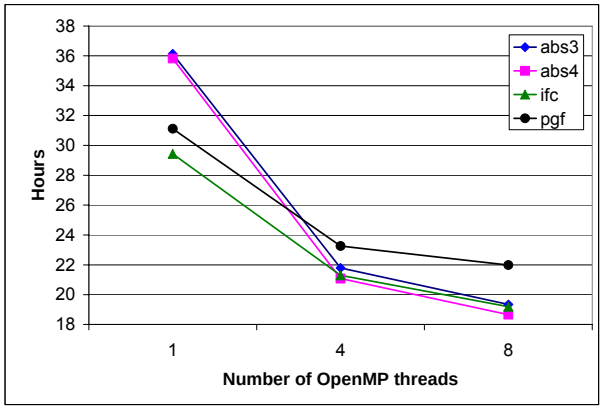

Fig 5.1: For the Intel platform with a thread count of 1, 4, and 8, this shows the OpenMP parallel runtime (in hours) for CMAQ 4.7.1 in the hybrid parallel version of the ROS3-HC solver for the Absoft (abs3, abs4), Intel (ifc) and Portland (pgf) compilers.

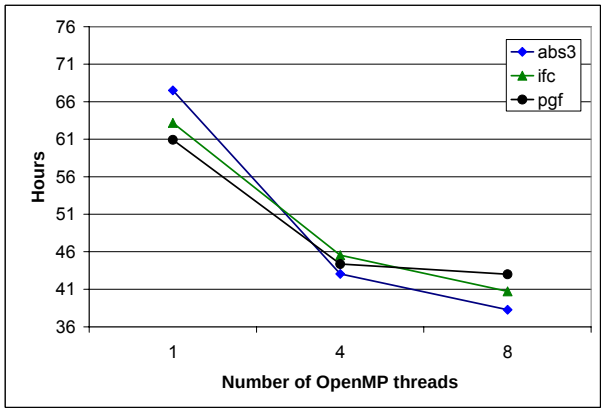

Fig 5.2: For the AMD platform with a thread count of 1, 4, and 8, this shows the OpenMP parallel runtime (in hours) for CMAQ 4.7.1 in the hybrid parallel version of the ROS3-HC solver for the Absoft (abs3), Intel (ifc) and Portland (pgf) compilers.

5.2 CMAQ OpenMP speedup vs U.S. EPA

In this section, and the next, two performance metrics are introduced to assess thread parallel performance in the ROS3-HC modified code:

(a) *Speedup* is the gain in runtime over the standard U.S. EPA runtime,
(b) *Scaling* is the gain in runtime for thread counts larger than 1, relative to the result for a single thread.

For the hybrid MPI+OpenMP modified CMAQ version with the Rosenbrock solver, Table 5.2 shows the speedup metric corresponding to the runtimes in Table 5.1. Figs 5.3 and 5.4, respectively, show the speedup results of Table 5.2 on the Intel and AMD platform for 1 to 16 cores. Note the doubling of the abscissa scale in Fig. 5.4 for the AMD case. This shows clearly the growing Absoft advantage with increasing thread count on both Intel and AMD platforms.

Table 5.2. OpenMP speedup of the hybrid MPI+OpenMP (ROS3-HC) version over the U.S. EPA release of CMAQ 4.7.1. The platforms are Intel and AMD for the Absoft (abs3, abs4), Intel (ifc) and Portland (pgf) compilers.

Compiler group	Platform	ROS3-HC Speedup by thread count				
		1	4	8	12	16
abs3	Intel	0.75	1.24	1.40		
abs4		0.72	1.23	1.39		
ifc		0.85	1.18	1.31		
pgf		0.88	1.18	1.25		
abs3	AMD	0.77	1.21	1.37	1.40	1.48
ifc		0.85	1.18	1.32	1.33	1.38
pgf		0.89	1.23	1.27		

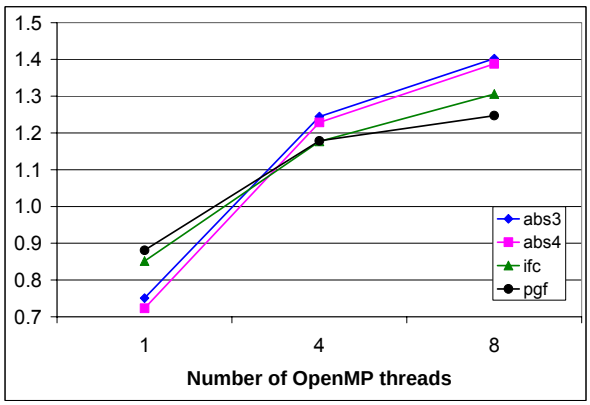

Fig 5.3: For the Intel platform with a thread count of 1, 4, and 8, this shows the OpenMP speedup of CMAQ 4.7.1 in the hybrid parallel (versus U.S. EPA) version of the ROS3-HC solver for the Absoft (abs3, abs4), Intel (ifc) and Portland (pgf) compilers.

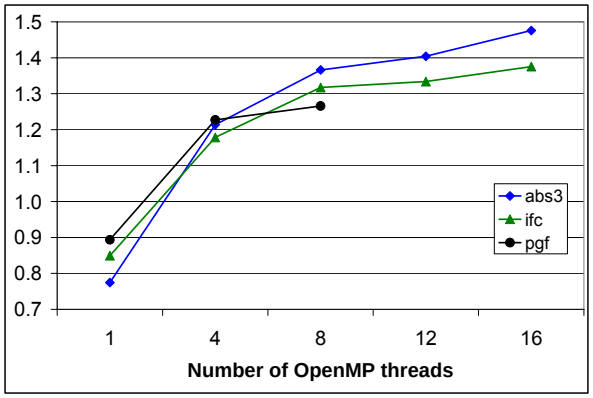

Fig 5.4: For the AMD platform with a thread count of 1, 4, 8, 12 and 16, this shows the OpenMP speedup of CMAQ 4.7.1 in the hybrid parallel (versus U.S. EPA) version of the ROS3-HC solver for the Absoft (abs3), Intel (ifc) and Portland (pgf) compilers.

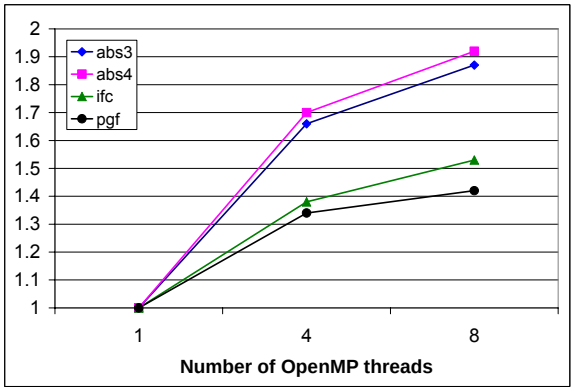

Fig 5.5: For the Intel platform with a thread count of 1, 4, and 8, this shows the OpenMP scaling for CMAQ 4.7.1 in the hybrid parallel version of the ROS3-HC solver for the Absoft (abs3∞, abs4), Intel (ifc) and Portland (pgf) compilers.

5.3 CMAQ 4.7.1 OpenMP scaling

For the hybrid MPI+OpenMP modified CMAQ version with the Rosenbrock solver, Table 5.3 shows the scaling metric corresponding to the runtimes in Table 5.1.

Table 5.3. OpenMP scaling in the hybrid MPI+OpenMP ROS3-HC version of the CMAQ 4.7.1 ROS3-HC solver on the HiPERiSM Intel and AMD platforms for the Absoft (abs3, abs4), Intel (ifc) and Portland (pgf) compilers.

Compiler group	Platform	ROS3-HC Scaling by thread count				
		1	4	8	12	16
abs3	Intel	1.00	1.66	1.87		
abs4		1.00	1.70	1.92		
ifc		1.00	1.38	1.53		
pgf		1.00	1.34	1.42		
abs3	AMD	1.00	1.57	1.76	1.81	1.91
ifc		1.00	1.39	1.55	1.57	1.62
pgf		1.00	1.37	1.42		

Figs 5.5 and 5.6 respectively, show OpenMP scaling results of Table 5.3 on the Intel and AMD platforms for 1 to 16 cores. Note the doubling of the abscissa scale in Fig. 5.6 for the AMD case. All three compilers show healthy increases in OpenMP scaling with increasing thread count. Here the higher scaling values for the Absoft case is a reflection of the longer runtime for a single thread.

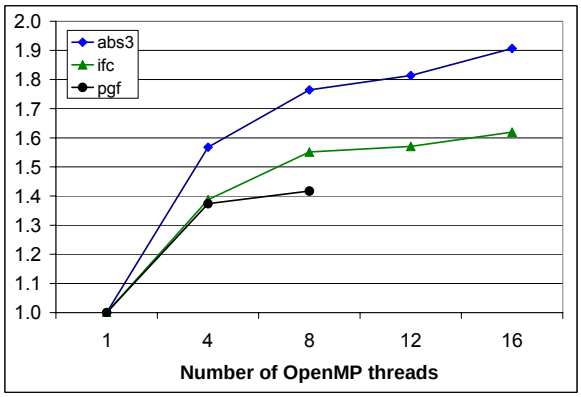

Fig 5.6: For the AMD platform with a thread count of 1, 4, 8, 12 and 16, this shows the OpenMP scaling for CMAQ 4.7.1 in the hybrid parallel version of the ROS3-HC solver for the Absoft (abs3), Intel (ifc) and Portland (pgf) compilers.

5.4 Saturation in OpenMP scaling

The results of the previous discussion demonstrate saturation in scaling as the thread count increases, as is particularly evident in Fig. 5.6. The origin of this is explained here using an adaptation of the argument given by R.W. Hockney [Hockney]. If runtime is T_1 for a single thread and T_P for p threads, then T_P has two time components. One from the Rosenbrock solver, T_{ROS}, and the other from serial work outside it, T_{SER}. Then $T_P = T_{SER} + T_{ROS} / p$, because p threads each concurrently performs 1/p of the work in the solver. Thread scaling, as used here, is the ratio $S_P = T_1/T_P = T_1/(T_{SER} + T_{ROS} / p)$. In the asymptotic limit, $p \rightarrow \infty$, scaling saturates at a maximum value of $S_\infty = T_1/T_{SER}$. From profiling results [Delic, 2009], one estimate of the ratio T_{SER}:T_{ROS} is 60:40, and then the saturation value is

S_∞ ~ 1.67. Some differences between compilers and platforms is to be expected because they perform differently in scheduling resources in serial and parallel code regions. This result is the so-called Amdahl saturation effect and is due to the presence of serial code outside the Rosenbrock solver that is executed with a single thread. The effect is evident in Table 5.3 and Figs. 5.5, 5.6, where the rate of scaling declines with thread counts above 4 at which point some 80% (or more) of the saturation value is reached. It is likewise reflected in the speedup comparison of Section 5.2, where the runtimes of two different algorithms are compared for the same problem size. As a footnote to this discussion, in an MPI implementation, the serial code part of CMAQ is substantially replicated for each MPI process and this results in redundant work that decreases MPI efficiency (as discussed in Section 4.3).

6. LESSONS LEARNED ABOUT CMAQ PERFORMANCE

6.1 Limits to CMAQ performance with MPI

CMAQ 4.7.1, with the Rosenbrock solver shows a decline in MPI parallel efficiency as the number of MPI processes increases. This results in an increase in processor idle time. The trend found here suggests that, in the U.S. EPA release, efficiency with hundreds of MPI processes would lead to average processor idle times of the order of 60%. Despite the reduction in runtime with increasing MPI process count, the focus in future research should be improved efficiency and enhanced CMAQ workload throughput. Confirmation of these observations is to be found in reports by others [Lee, 2010] that increasing the number of MPI processes beyond 100 does lead to an increase in CMAQ runtime.

6.2 Compilers for the CMAQ model

Differentiating compilers for CMAQ based on performance alone is now more difficult because the performance window has narrowed between the available choices as compiler vendors have evolved to embrace new architectures.

This comparison of three vendor compilers shows that while each offers value for CMAQ performance there is little justification in limiting the standard distribution of CMAQ to only one or two choices. Specifically the results for the Absoft compiler suggest that is should be included in future distributions of CMAQ.

An additional discovery at HiPERiSM Consulting, LLC, has been the consequences compiler optimization choices have for numerical precision. The highest level optimizations generally have a reduced accuracy for some CMAQ species concentrations. For this reason care needs to be exercised in the use of compilers to avoid erroneous model predictions.

6.3 CMAQ in multi-thread mode

This analysis compared runtime of CMAQ 4.7.1 in the hybrid MPI and OpenMP parallel version with the U.S. EPA release. The observations indicated that the multi-threaded speedup:

- Showed a range of 1.25 to 1.4 with 8 parallel threads and values as high as 1.48 for 16 threads.
- Saturates with asymptotic thread count due to the remaining serial work outside of the Rosenbrock solver.
- Does not depend on the choice of hardware since both AMD and Intel platforms showed similar trends.
- Was Independent of compiler choice based on comparison of compilers from Absoft, Intel and the Portland Group.

6.4 Comparing hardware platforms

There is a large difference in runtime between AMD and Intel platforms for the same model simulation of an individual serial run. However, with more cores on the AMD platform there is more flexibility and greater workload throughput capability. This is explained in a cost benefit analysis of these two hardware options for CMAQ workloads in a recent report [Delic, 2011].

7. CONCLUSIONS

This report has described an analysis of CMAQ 4.7.1 behavior in the standard U.S. EPA release (with MPI only) and a parallel hybrid (OpenMP and MPI) version of CMAQ for the Rosenbrock solver. Good speedup trends result from an increase in the number of parallel threads. This trend was observed with three compilers on two hardware platforms. However, limits were observed to performance opportunities with an MPI (only) implementation of the standard U.S. EPA release. The limitations were due to increasing average processor idle time during the course of a simulation as the MPI process count increased.

Compilers from Absoft, Intel and the Portland group all offered value for the CMAQ model and the current overall performance leader is the compiler from Absoft.

Further opportunities remain for thread parallelism in other parts of the CMAQ model outside of the solver and work in this direction continues at HiPERiSM Consulting, LLC. In view of the limitations with MPI evolution this effort is directed at performance with GPGPU technology [CUDA] where using thousands of threads for CMAQ simulations brings its own challenges.

REFERENCES

ABSOFT: The Absoft Corporation
http://www.absoft.com

CMAS: http://www.cmascenter.org/

CUDA:
http://www.nvidia.com/object/cuda_home_new.html

Delic, 2003-2010: see presentations at the Annual CMAS meetings (http://www.cmasecenter.org).

Delic, 2011: Technical Report HCTR-2011-4 at http://www.hiperism.com

Hockney, R.W., *The Science of Computer Benchmarking*, SIAM, Philadelphia, 1996.

INTEL: Intel Corporation, http://www.intel.com

Lee, Pius, *et al.,* 2010, 9th Annual CMAS Conference, Chapel Hill, NC, October 11-13, 2010.

PGI: The Portland Group http://www.pgroup.com