Presented at the 8th Annual CMAS Conference, Chapel Hill, NC, October 19-21, 2009

ANALYSIS OF CMAQ PERFORMANCE ON QUAD-CORE PROCESSORS

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

1. INTRODUCTION

CMAQ offers a choice of three gas chemistry solvers: Euler-Backward (EBI), Rosenbrock (ROS3), and Gear (GEAR). However there is an expansion in computational (wallclock) time relative to the faster EBI solver with the use of either ROS3 or GEAR solvers. The respective expansion factors are 1.6 and 2.2. Therefore, for greater throughput, the EBI solver is generally used in model simulations. Nevertheless, both ROS3 and GEAR, with proper tolerances set, are more accurate algorithms for gas chemistry modeling. They also offer potential for parallel code development and vector operations which are lacking in the EBI algorithm. These considerations motivated HiPERiSM Consulting, LLC to develop a thread-parallel version of the Rosenbrock solver (hereafter ROS3-HC) that completes the same simulation in a wall-clock time comparable to that of the EBI solver for the same gas chemistry.

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 new (inner) parallel layers developed at HiPERiSM have added both thread-level parallelism and instruction-level parallelism (at the vector loop level). The inclusion of these two additional levels of parallelism in the Rosenbrock solver enables sufficient scaling with both the number of threads and vector length to complete the same 24 hour simulation in a wall clock time comparable to that for the EBI solver. For the results reported here CMAQ was executed in serial mode (i.e. no MPI) on all platforms. The serial results reported here will propagate to each MPI process and also enhance performance of the MPI implementation of CMAQ. The 3-level hybrid parallel MPI-thread-vector implementation of ROS3-HC will be the subject of a separate study once thread and vector performance tuning has been completed.

On the new generation of quad-core processors the ROS3-HC parallel solver significantly reduces the wall-clock time of large-scale CMAQ model simulations compared to the standard U.S. EPA distribution. The present study was undertaken to identify how performance is enhanced and how numerical results compare with those of the standard U.S. EPA distribution. The following sections present the details of this study as a follow-on to the previous year's report for the EBI solver (Young and Delic, 2008).

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

2. CHOICE OF HARDWARE AND OPERATING SYSTEM

CMAQ 4.6.1 was compiled with a new release of the Intel™ Fortran compiler on 64-bit Linux operating systems. The hardware systems chosen were the three platforms at HiPERiSM Consulting, LLC, shown in Table 2.1.

Table 2.1. Platforms at HiPERiSM Consulting, LLC

Platform	Itanium	64EMT	quad-core
Processor	Intel™ IA64 (107W)	Intel™ IA32 64EMT	Intel™ IA32 (X5450)
CPUs/core	single CPU	dual CPU	quad-core
Clock	1.5GHz	3.4GHz	3.0GHz
Bandwidth	6.4GB/sec	6.4GB/sec	10.6GB/sec
Bus speed	400 MHz	800 MHz	1333 MHz
L1 cache	32KB	32KB	64 KB
L2 cache	1 MB	1MB	12MB
L3 cache	4MB	NA	NA

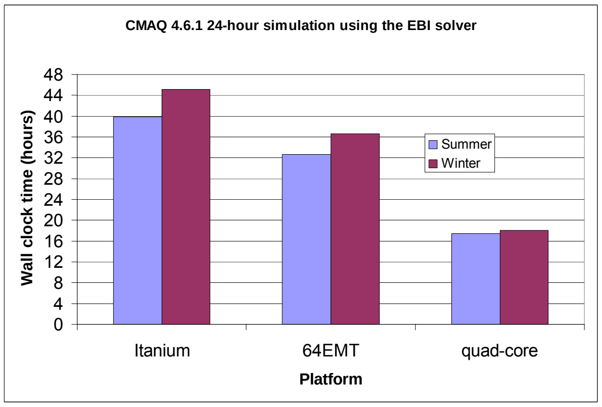

Fig 2.1: Time to solution with the serial version (no MPI) for the U.S. EPA distribution of CMAQ 4.6.1 using the EBI solver on three platforms for two 24-hour episodes.

Clock rate ranged from 1.5GHz to 3.4 GHz. On all platforms L1 cache is divided equally between data and instructions. Three important features to note are the increased L1, L2 cache sizes, and peak bandwidth on the quad-core platform compared to the other two. These features alone are expected to have a very significant impact on CMAQ performance because of the heavy memory footprint it has (Young and Delic, 2008, and Delic, 2003-2006). This is confirmed by the results shown in Fig. 2.1 for the EBI solver. For later reference, times for the Summer episode are: 39.9, 32.6, and 17.4 hours, respectively. Note the sharp improvement on the quad-core platform versus the earlier generation 64EMT and Itanium (SGI Altix) platforms.

3. EPISODES STUDIED

The model episodes selected for this analysis were for January 10 and August 14, 2006 (hereafter Winter and Summer, respectively). Both used the CB05 mechanism with Chlorine extensions and the Aero 4 version for PM modeling. For all results reported here the ROS3 solver was used for the gas chemistry. Both episodes were run for a full 24 hour scenario on a 279 X 240 Eastern US domain at 12 Km grid spacing and 34 vertical layers. This results in a total of 2,276,640 grid cells that CMAQ subdivides into blocks through a BLKSIZE parameter to enhance performance in the chemistry solver.

4. PARALLEL ROSENBROCK SOLVER

4.1 Parallel code development

A typical ROS3 runtime profile for the standard U.S. EPA distribution of CMAQ 4.6.1 is shown in Table 4.1 and this summarizes the proportion of the total runtime for the gas chemistry solution.

Table 4.1. Relative proportion of total runtime on the quad-core for the ROS3 solver in the U.S. EPA distribution of CMAQ 4.6.1 for the two episodes.

Function	Summer		Winter	
	%	Σ%	%	Σ%
rbdecomp	16.5	16.5	15.0	15.0
rbsolve	10.2	26.7	10.5	25.5
rbfeval	7.6	34.3	7.9	33.4
rbjacob	6.5	40.8	6.7	40.1
rbsolver	3.3	44.1	3.4	43.5

Since the ROS3 solver accounts for almost half of the total runtime an important motivation for this work was to reduce the time required for the gas chemistry solution. As an example, achieving a goal of an order of magnitude reduction in the solver part would significantly shrink the runtime expansion factor of the ROS3 solver versus the EBI solver. This goal is achieved by modifying the Rosenbrock procedures used in the standard U.S. EPA distribution listed in the “EPA” column of Table 4.2.

Table 4.2. Subroutine procedures in the U.S. EPA and HC versions of the ROS3 solver.

EPA	HC
GRID_CONF	GRID_CONF-hc
rbdriver	rbdriver-hc
rbdata_mod	rbdata_mod-hc
rbdecomp	NA
rbfeval	NA
rbinit	rbinit-hc
rbjacob	NA
rbsolve	NA
rbsolver	rbsolver-hc
rbsparse	rbsparse-hc

The “HC” column of Table 4.2 shows the subroutines used in HiPERiSM version and entries marked NA no longer appear in the call tree because they have been assimilated into subroutine rbsolver-hc. The ROS3-HC version was created by successive code structure modifications of the standard ROS3 solver without changing the science of the model in any way. The modified ROS3-HC applies a thread parallel strategy that has three prongs:

1. Storage of concentrations of all gas chemistry species in a global array with the first index of dimension BLKSIZE in the solver.
2. Partitioning of this index into BLKSIZE/NCMAX chunks for distribution to separate threads in a parallel thread team.
3. Organization of thread private arrays for each thread into vector loops with maximum range NCMAX.

Specific restructuring steps applied to the standard CMAQ gas chemistry solver included:

- Manual inline of repeated procedure calls (specifically all subroutines listed as NA in Table 4.2)
- Reordering of indices on key arrays referenced in thread parallel regions and vector loops (this required modifications in subroutines rbdata_mod and rbinit where arrays are defined and allocated)
- Arrangement of loops so that the primary one is always on the cell index as the target for thread tasking (with range 1 to BLKSIZE in steps of NCMAX)
- Arrangement of secondary loops so that they target SSE vector instructions (vector loop index with range 1 to NCMAX).

- Value assignment for BLKSIZE and NCMAX in module GRID_CONF such that mod(BLKSIZE, NCMAX) is zero.
- Declaration of thread parallel regions by insertion of OpenMP directives and classification of local (thread private) and global (shared) variables.
- Simplification/streamlining of redundant code in subroutine rbsparse to improve ROS3 solver performance.

This thread-vector parallel strategy can only succeed if there is sufficient coarse grain parallel work for each thread. To explore both parallel and vector granularity the code in rbsolver-hc has been arranged to include two user-selectable parameters BLKSIZE and NCMAX. An empirical study is required to determine both the extremes and the optimal values of BLKSIZE and NCMAX for specific platforms. Table 4.3 summarizes all choices studied in this analysis and shows the number of parallel chunks available for each value of BLKSIZE/NCMAX. Note the consequences for the number of CMAQ grid cell blocks that need to be processed in the episodes studied here.

Table 4.3. Choices for BLKSIZE and NCMAX parameters explored in this analysis.

BLKSIZE	NCMAX	No. of parallel chunks	No of grid cell blocks
48	24	2	47430
48	12	4	47430
64	32	2	35572.5
64	16	4	35572.5
80	40	2	28458
80	20	4	28458
96	48	2	23715
96	24	4	23715
96	12	8	23715
144	72	2	15810
144	36	4	15810
144	18	8	15810
240	120	2	9486
240	60	4	9486
240	30	8	9486
480	240	2	4743
480	120	4	4743
480	60	8	4743
480	30	16	4743
1440	360	4	1581
1440	180	8	1581
1440	120	12	1581
1440	90	16	1581
1440	45	32	1581
1440	20	72	1581

By varying the BLKSIZE and NCMAX parameters, runtime scaling behavior may be studied. Parallel inefficiencies do result at runtime for poor parameter choices at two extremes. Values that are too small cause the domination of parallel thread overhead and inefficiencies due to short vector lengths. Alternatively, values that are too large result in elevated cache misses and escalation in total memory size. At the too-large extreme either CPU cores are idled while waiting on data from memory (when it is not found in cache), or the executable gives a segfault when loaded (due to insufficient memory).

4.2 Thread memory considerations

An important consideration when creating thread parallel code is the cost of increased memory usage because each thread has it own copy of thread private (or local) variables. In the ROS3-HC version of CMAQ the total memory used by local variables is shown in Fig. 4.1. Local memory usage scales with thread count and choice of the BLKSIZE and NCMAX combination. Since the 12MB L2 cache on the quad-core platform is shared by all threads, there is a performance penalty for choices that cause cache use inefficiency. In the parallel region it is the choice of NCMAX that dominates the memory demand: choosing smaller values decreases the memory required and also increases the number of parallel chunks available for parallel thread team processing.

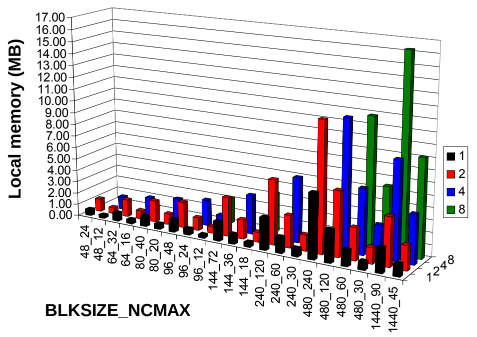

Fig 4.1: In the parallel thread region the amount of local memory (that is private to each thread) increases as the thread count increases (shown from front to back in this figure) and also as the choice of BLKSIZE and NCMAX increases (shown from left to right in this figure).

In the ROS3-HC parallel regions, threads also access global memory allocated for shared variables (such as concentration arrays). Fig. 4.2 shows how this memory demand scales with BLKSIZE and NCMAX. From Fig. 4.2 it is clear that the global memory demand is dominated by the choice of BLKSIZE.

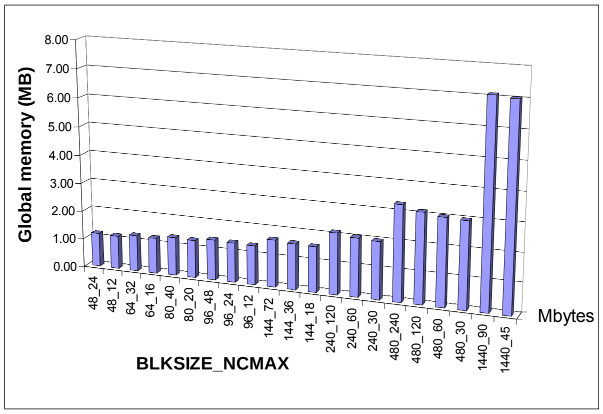

Fig 4.2: In the parallel thread region the amount of global memory (that is shared by all threads) increases as the choice of BLKSIZE changes (shown from left to right in this figure).

5. THREAD PARALLEL PERFORMANCE

5.1 Runtime versus vector length NCMAX

The U.S. EPA distribution of the ROS3 solver has relatively few vector loops. Invoking vector (SSE) instructions on IA32 architectures is important for efficient runtime performance of models. For this reason a special effort was made to ensure all possible inner loops were vector candidates in ROS3-HC. Tests with the solver in ROS3-HC at HiPERiSM Consulting, LLC, have shown that the Intel™ compiler will vectorize a different number of loops depending on the choice of the vector length, NCMAX. As examples, inside the thread parallel regions constructed in rbsolver-hc, with NCMAX=4, only 27 loops will vectorize. On the other hand with NCMAX=12, all 43 candidate loops vectorize. For this reason NCMAX is chosen to be not less than 12 with the upper limit dependent on the choice of the BLKSIZE parameter. Table 4.3 lists the full range of parameter values explored in this study.

5.2 Runtime versus block size BLKSIZE

The number of cells in a block is defined by the parameter BLKSIZE in CMAQ and this value is hard-coded in module GRID_CONF. CMAQ is distributed with a fixed value of BLKSIZE=50 and, in general, variations to match cache size on different platforms is recommended, even for the standard CMAQ distribution. However, the variations discussed here are specific to the thread-parallel performance of ROS3-HC. Clearly, BLKSIZE cannot be set to the total number of grid cells (= 2,276,640 in the episodes studied here) because of memory limitations. Ideally, BLKSIZE should divide the total number of grid cells without a remainder, and also must be divisible by NCMAX without remainder. In the examination of ROS3-HC performance a range of values for BLKSIZE from 48 to 1440 was explored. Wall clock time depends on both NCMAX and BLKSIZE values. Figs. 5.1, 5.2 show the variation in time to solution for ROS3-HC (in hours) as a function of these respective parameters. The respective groups of points show the times for 1, 2, 4, and 8 parallel threads (as defined in the legend). In Fig. 5.1 the vertical stacking is due to the same value of NCMAX occurring for different BLKSIZE parameter choices. The converse is the case in Fig. 5.2.

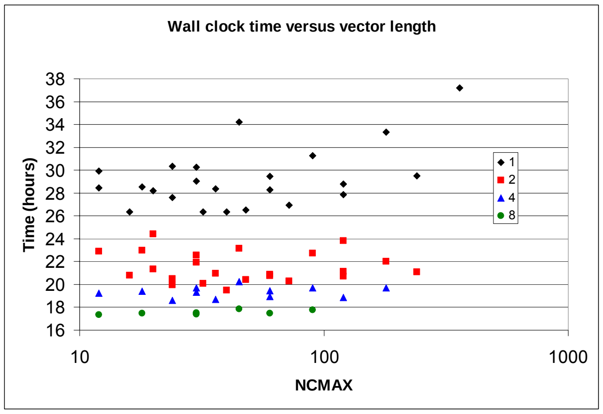

Fig 5.1: For thread parallel execution with 1, 2, 4, and 8 threads this shows the wall clock time as a function of the choice for NCMAX (the vector length in the parallel region).

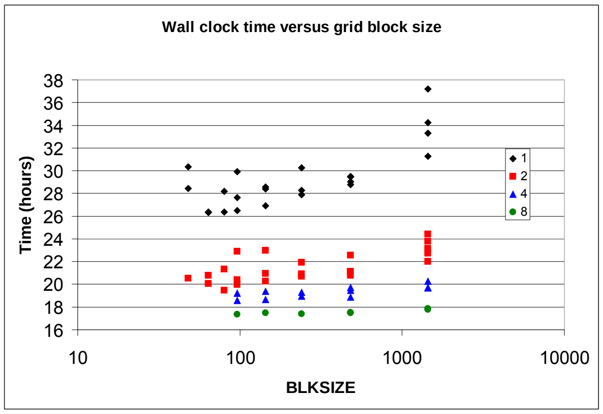

Fig 5.2: For thread parallel execution with 1, 2, 4, and 8 threads this shows the wall clock time as a function of the choice for BLKSIZE.

5.3 Thread parallel scaling

For a subset of BLKSIZE and NCMAX values from Table 4.3 that allow at least 8 chunks, Table

5.1 shows parallel speedup with increasing thread count (from 1 to 8) in ROS3-HC.

Table 5.1. Speedup for the summer episode.

BLKSIZE_NCMAX	Number of threads			
	1	2	4	8
96_12	1.00	1.31	1.56	1.73
144_18	1.00	1.24	1.47	1.63
240_30	1.00	1.38	1.57	1.74
480_30	1.00	1.28	1.47	1.66
480_60	1.00	1.41	1.51	1.68
1440_45	1.00	1.47	1.68	1.92
1440_90	1.00	1.38	1.59	1.76

The corresponding wall clock times are listed in Table 5.2 and Fig. 5.3 shows these in a perspective plot. Note that the results for 8 threads in Table 5.2 are close to the time of 17.4 hours for the EBI solver in Fig. 2.1.

Table 5.2. Summer episode wall clock time (hours)

BLKSIZE_NCMAX	Number of threads			
	1	2	4	8
96_12	29.9	22.9	19.2	17.3
144_18	28.6	23.0	19.4	17.5
240_30	30.2	21.9	19.3	17.4
480_30	29.0	22.6	19.7	17.5
480_60	29.4	20.8	19.5	17.5
1440_45	34.2	23.2	20.3	17.8
1440_90	31.3	22.7	19.7	17.8

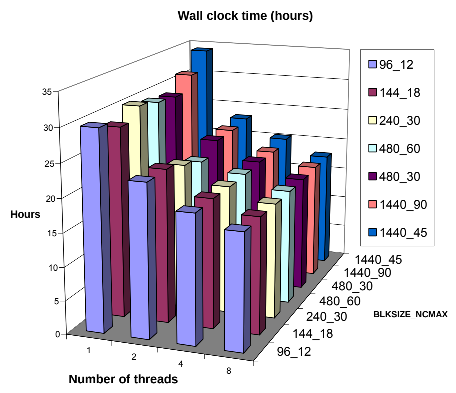

Fig 5.3: For thread parallel execution with 1, 2, 4, and 8 threads, respectively, this shows the wall clock time as a function of BLKSIZE and NCMAX parameter choices.

As the thread count increases Fig. 5.3 clearly shows that (a) the wall clock time decreases, and (b) the variability in wall clock time as a function of BLKSIZE and NCMAX parameters reduces sharply. The latter result is a cache and memory related effect and shows that partitioning of global arrays amongst threads reduces the negative consequences of cache use inefficiency. In other words: with a larger number of parallel threads the choice of BLKSIZE and NCMAX values is not critical to runtime performance.

5.4 ROS3-HC versus U.S. EPA CMAQ

The HiPERiSM Consulting, LLC, version of the Rosenbrock solver is compared against the U.S. EPA version in Table 5.3 for identical compiler options. For a single thread, the ROS3-HC runtime is larger than the U.S. EPA version. However, the ROS3-HC version has a speedup in the range 1.2 to 1.7 as the number of threads climbs from 2 to 8. In view of the observations in the previous section and the single thread result here, ROS3-HC should be used with 2 or more threads. As a final observation, the best run times found for the EBI solver were between 16.1 and 17.4 hours (depending on the choice of compiler options) and therefore the ROS3-HC solver with 8 threads is within 8% of the best EBI solver run time.

Table 5.3. Summer episode wall clock time (hours) and speedup with ROS3-HC for BLKSIZE=480, NCMAX=60 on the quad-core platform.

CMAQ version for Rosenbrock solver	Number of threads			
	1	2	4	8
U.S. EPA (hours)	25.3	NA	NA	NA
ROS3-HC (hours)	29.4	20.8	19.5	17.5
ROS3-HC (speedup)	0.86	1.22	1.51	1.68

6. NUMERICAL RESULTS

Numerical differences for concentration values predicted by the standard U.S. EPA and HiPERiSM Consulting, LLC, versions of the ROS3 solver were examined in great detail. Software utilities were developed to read CMAQ output files for concentration values of all 98 species, in all grid cells, at all time steps, for a single layer from the respective output files (with approximately 1.6 million values compared per layer). Automated procedures generate both scatter plots and statistical summaries of differences for each of the 98 species. The comparisons are repeated for any combination of thread count, BLKSIZE, and NCMAX parameter values in ROS3-HC. In comparing concentration values it should be noted that CMAQ applies an L_2 norm over species values in the convergence criterion for gas chemistry. This metric will not minimize the maximum error. Therefore values for individual species may exceed this default error tolerance. Nevertheless, visual inspection of scatter plots (not included here for space reasons) showed no differences with respect to changes in thread count, BLKSIZE, or NCMAX values.

Table 6.1: Some statistics for comparison of selected concentration predictions between U.S. EPA standard and HiPERiSM thread parallel versions of the Rosenbrock solver in CMAQ. For this table ROS3-HC used BLKSIZE=144, NCMAX=18 and 8 threads. All layer 1 concentrations were compared (more than 1.6 million values). The units are ppmV (O3 to NH3) and micrograms/m^3 (all other species shown).

Species name	Absolute error			Coefficient of variation
	max	mean	stdev	
O3	3.2e-5	5.3e-8	1.1e-7	2.1
NO	2.2e-5	8.2e-9	9.0e-8	11.0
NO2	2.2e-5	1.1e-8	9.3e-8	8.6
NO3	4.4e-8	7.6e-11	3.1e-10	4.1
CO	2.9e-6	1.5e-7	2.1e-7	1.4
SO2	2.4e-5	1.3e-9	2.3e-8	18.2
PAR	6.6e-7	2.3e-8	3.6e-8	1.6
ISOP	2.6e-7	3.7e-10	1.1e-9	3.1
NH3	2.2e-4	7.3e-9	4.7e-7	64.5
ANO3I	2.0e-2	2.1e-7	4.5e-5	215.0
ANO3J	5.6e-1	5.2e-6	1.1e-3	222.6
ANH4I	6.7e-3	2.8e-7	1.5e-5	54.6
ANH4J	1.6e-1	3.6e-6	3.3e-4	93.5
ASO4I	2.5e-2	1.2e-6	6.2e-5	50.2
ASO4J	7.8e-2	4.2e-6	9.9e-5	23.4
A25J	9.1e-3	8.0e-7	1.2e-5	14.4
AECI	2.7e-5	1.2e-9	5.8e-8	50.0
AECJ	1.7e-3	1.2e-7	2.3e-6	18.4

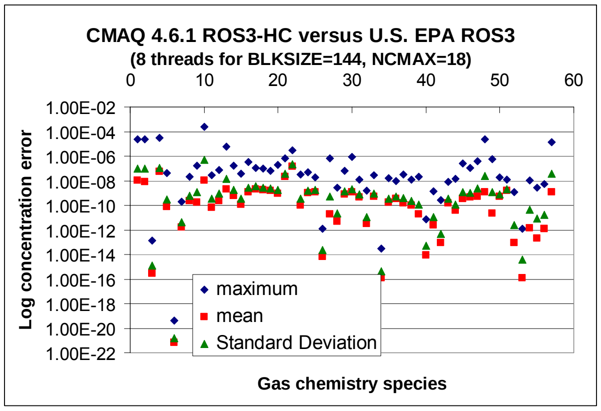

Fig 6.1: Absolute error statistics for 58 gas chemistry species concentration comparisons between U.S. EPA standard and HiPERiSM thread parallel versions of the ROS3 solver in CMAQ. All layer 1 species concentrations were compared in units of ppmV.

For a selection of species, Table 6.1 shows descriptive statistics in a comparison of over 1.6 million individual concentration values in Layer 1 (of the 34 grid layers) in the simulation for the Summer episode. Similar results are observed for the Winter episode. It is concluded that numerical differences observed between the U.S. EPA and HiPERiSM Consulting, LLC, versions of the ROS3 solver are below the error tolerance threshold used in CMAQ. Fig. 6.1 shows some statistics for 58 gas chemistry concentration predictions when both versions of the ROS3 solver are compared. Typically, the comparisons for gas chemistry species concentrations agree to better than the default RTOL value 1.0e-03 for all species concentrations in layer 1.

7. CONCLUSIONS

7.1 Parallel execution of ROS3-HC

This report has described a thread-parallel version of CMAQ with the Rosenbrock solver that:

- Ameliorates code inefficiencies such as:
  - Insufficient use of vector instructions
  - Excessive control transfer instructions
  - Inefficient memory access.
- Delivers parallel performance with 8 threads on quad-core processors that is:
  - Close to the U.S. EPA EBI solver runtime
  - Is 1.7 times faster than the U.S. EPA standard distribution.
- Predicts concentration values that are within the CMAQ convergence criteria.

7.2 Next steps

Further opportunities abound for performance enhancement of CMAQ through thread parallelism and porting to new hardware technologies.

Specifically, next steps at HiPERiSM will:

- Tune runtime multithread performance to reduce cache use inefficiency.
- Seek opportunities higher up the CMAQ call tree for thread parallelism and enhanced vector performance for data locality.
- Port to general purpose graphical processor unit (GPGPU) technology.
- Compare performance with legacy technology such as the SGI Altix.
- Include benchmarks with multiple MPI processes
- Continue statistical analyses of concentration value comparisons with the standard U.S. EPA distribution to validate the code modifications.
- Apply similar enhancements to the GEAR solver in CMAQ.

REFERENCES

Delic, 2003-2006: see presentations at the Annual CMAS meetings.

Young and Delic, 2008, 7th Annual CMAS Conference, Chapel Hill, NC, October 6-8, 2008.