Presented at the 15th Annual CMAS Conference, Chapel Hill, NC, October 24-26 2016

A THREAD PARALLEL SPARSE CHEMISTRY SOLVER FOR CMAQ 5.1

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

1. INTRODUCTION

This presentation reports on implementation of the parallel sparse matrix solver, FSparse, in the Chemistry Transport Model (CTM) in CMAQ [1]. This is applicable in the CMAQ version that uses either the Rosenbrock (ROS3) or SMV Gear (GEAR) algorithms in the CTM. In this report results of thread parallel versions of the EBI and ROS3 solvers is presented while the GEAR algorithm is deferred for future study. In FSparse different blocks of cells are distributed to separate threads in the parallel thread team. Along the way some bugs and peculiarities in the CMAQ code interaction with the compiler of choice were uncovered. Species concentration values are compared for original and FSparse methods using VERDI and inspection of residual norms in the sparse solver. Some comments on numerical analysis is presented based on the comparisons.

2. TEST BED ENVIRONMENT

2.1 Hardware

The hardware systems chosen were the platforms at HiPERiSM Consulting, LLC, shown in Table 2.1. Each of the two nodes host two Intel E5v3 CPUs with 16 cores each. Each node has, in addition, four Intel Phi co-processor many integrated core (MIC) cards with 60 and 59 cores for the respective models. With four MIC cards per node, and up to 4 threads per MIC core, the total thread count is 960 and 944, respectively. This combination allows for testing of hybrid parallel versions of CMAQ on either host or first generation Intel Phi processor [2]. In the latter case the thread parallel region of the CTM is offloaded to the Phi processors.

2.2 Compiler

This report implemented the Intel Parallel Studio® suite (release 16.0) using options for either host CPU or Phi coprocessor. The latter required code modification to identify MIC attributes within a single source code. The extensive reporting options were used to investigate optimization effectiveness.

2.3 Episode studied

This report used the benchmark test data available in the CMAQ 5.1 download. This model episode was for July 1st, 2011, using the cb05e51_ae6 mechanism with 147 active species and 343 reactions. For day/night chemistry this results in 1224/1158 non-zero entries in the Jacobian matrix. The episode was run for a full 24 hour scenario on a 100 X 72 California domain at 12 Km grid spacing and 35 vertical layers for a total of 252,000 grid cells. This domain is some ten times smaller than that reported previously in [1]. In this report a variable number of MPI processes (NP) were used in the EPA version of CMAQ and only NP=1 in the OpenMP version.

Table 2.1. CPU platforms at HiPERiSM Consulting, LLC

Platform	Node20	Node21
Operating system	SuSE Linux 13.2	SuSE Linux 13.2
Processor	Intel™ IA32 (E5-2698v3)	Intel™ IA32 (E5-2698v3)
Coprocessor	4 x Intel Phi 7120	4 x Intel Phi 5110
Peak Gflops (SP/DP)	589 (SP)	589 (SP)
Power consumption	135 Watts	135 Watts
Cores per processor	16	16
Power per core	8.44 Watts	8.44 Watts
Processor count	2	2
Total core count	32	32
Clock	2.3 GHz	2.3 GHz
Bandwidth	68 GB/sec	68 GB/sec
Bus speed	2133 MHz	2133 MHz
L1 cache	16x32 KB	16x32 KB
L2 cache	16x256 MB	16x256 KB
L3 cache	40 MB	40 MB

In the following two performance metrics are introduced to assess thread parallel performance in the OpenMP modified code:

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

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

3. RESULTS FOR THE STANDARD MODEL

3.1 EBI solver on host

This section summarizes results of the standard CMAQ 5.1 distribution in the testbed environment identified in Section 2. The optimization level was "-O2" because higher optimizations caused segmentation faults (segfaults) at runtime. The combination of MPI processes, NP = NPROW x NPCOL, are those shown in Table 3.1 for the EBI solver algorithm.

Table 3.1. Wall clock times (in seconds) for the U.S. EPA EBI version of CMAQ on Intel host CPUs.

NPROW X NPCOL	EBI			
	Value of NPCOL			
	1	2	4	8
1	10733			
2	5601			
4	3128	3103		
8	1916	1854	1858	
12		1422		
16	1309	1220	1189	1188
24		966	931	
32	916		788	771

Fig. 3.1 shows that, while the wall clock time declines, efficiency also decreases for higher MPI process counts. This is more clearly reflected in Fig. 3.2 which shows the parallel efficiency as a function of increasing MPI process count. Parallel efficiency is the speedup versus 1 MPI process divided by the the number of MPI processes. This is below 50% with more than 16 MPI processes and as low as 37% with 32 MPI processes, i.e. host cores are idle for as much as two thirds of the wall clock time.

3.2 All CTM solvers on host

For all available solvers the standard model was compiled with "-O1" because again segfaults occurred when higher optimizations were used for ROS3 or GEAR solver versions of CMAQ.

A breakdown of where time is expended in CMAQ 5.1 science processes is shown in Fig. 3.3 for GEAR, ROS3, and EBI solver algorithms. The relative fraction of total wall clock time is shown as a percentage in Fig. 3.4.

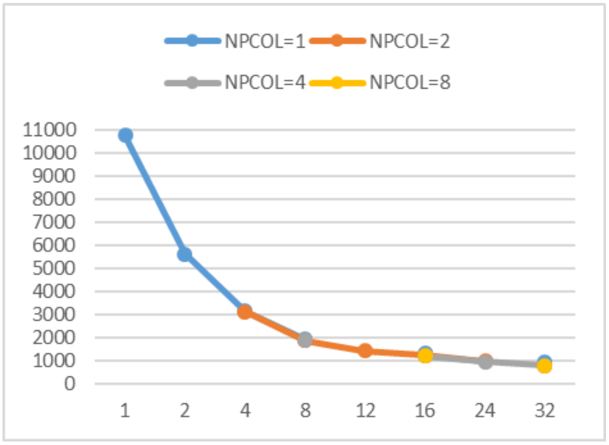


Fig 3.1: Wall clock time from Table 3.1 versus number of MPI processes for the EBI solver version.

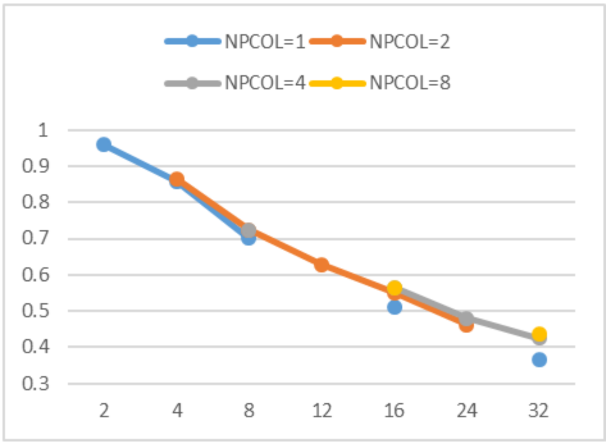


Fig 3.2: MPI parallel efficiency versus number of MPI processes for the EBI solver version

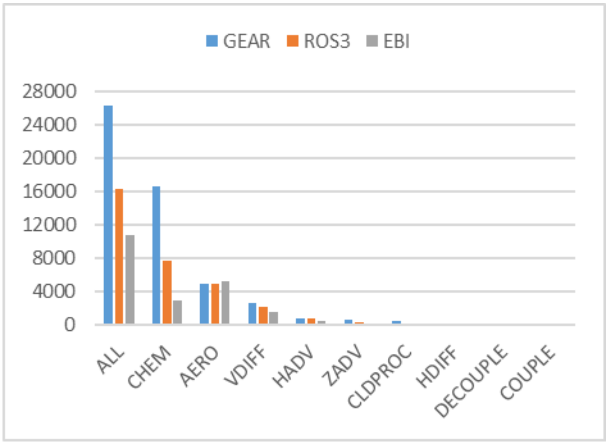


Fig 3.3: Wall clock time by science process for EBI, ROS3, and GEAR versions of CMAQ for NP=1.

The totals of wall clock time for each version with various values of NP is shown in Table 3.2.

Obviously, wall clock time increases as the solver changes from EBI, to ROS3 and then GEAR. For example, with NP=16 the ratio in time is 1:1.5:1.9. However, as shown in Fig. 3.5, the parallel efficiency declines to ~70% when NP=16, and ~30% when NP=64. This loss in parallel efficiency is due to the diminished work load per MPI process with a domain of 252,000 cells. When partitioned amongst the available number of MPI processes, after division into blocks of 50 cells: 252,000/50 = 5040 blocks for NP = 1, and 5040 / NP thereafter, when NP > 1.

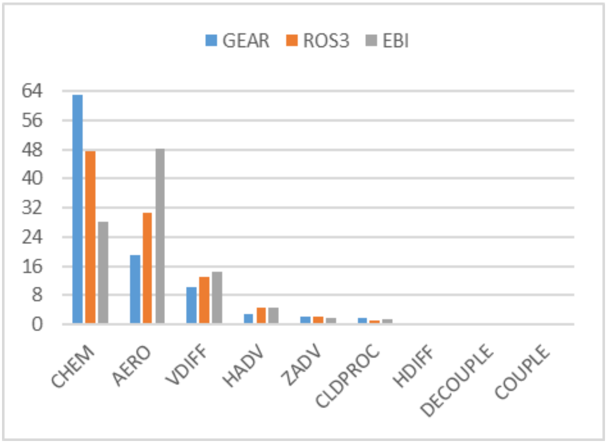


Fig 3.4: Fraction of wall clock time (percent) by science process for EBI, ROS3, and GEAR versions of CMAQ for NP=1. Note that CHEM is not the dominant process for the EBI case.

Table 3.2. Wall clock times (in seconds) for the U.S. EPA version of CMAQ on Intel host CPUs.

NPROW X NPCOL	CTM solver algorithm		
	EBI	ROS3	GEAR
1	11881	16350	25229
4	3338	4790	7788
16	1164	1508	2239
64	624	941	1180

3.3 MPI performance on host

Table 3.3. Wall clock times (in seconds) for the U.S. EPA version of CMAQ on Intel host CPUs showing fraction of time in MPI calls.

NPROW X NPCOL	Model versus MPI time		
	EBI code	MPI	Ratio of MPI time
2	1.08e+04	0.53e+03	4.6%
4	1.23e+04	1.11e+03	8.2%
8	1.31e+04	2.03e+03	13.4%
16	1.66e+04	6.69e+03	28,7%
32	2.56e+04	1.57e+04	38.0%

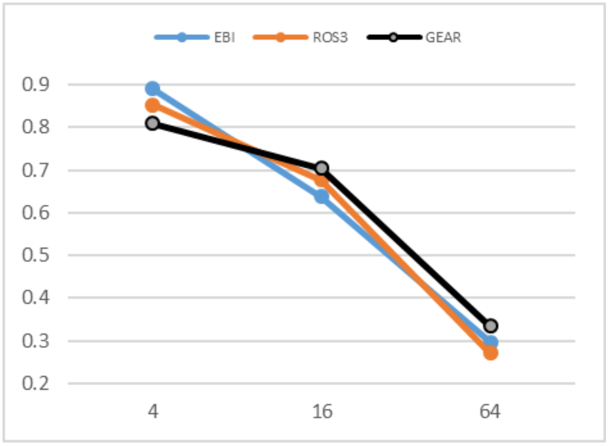
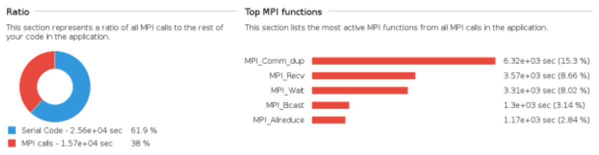


Fig 3.5: Parallel efficiency versus number of MPI processes for all three solver algorithms.

The VTune performance analyzer was used to study the MPI performance of the EPA version of CMAQ 5.1. Table 3.3 shows the increasing fraction of wall clock time spent in MPI procedures to demonstrate another factor in diminishing parallel efficiency. Fig. 3.6 shows the top MPI functions for the case of 32 MPI processes.



Ratio

This section represents a ratio of all MPI calls to the rest of your code in the application.

🟦 Serial Code - 2.56e+04 sec 61.9 %
🟥 MPI calls - 1.57e+04 sec 38 %

Top MPI functions

This section lists the most active MPI functions from all MPI calls in the application.

MPI_Comm_dup	6.32e+03 sec (15.3 %)
MPI_Recv	3.57e+03 sec (8.66 %)
MPI_Wait	3.31e+03 sec (8.02 %)
MPI_Bcast	1.3e+03 sec (3.14 %)
MPI_Allreduce	1.17e+03 sec (2.84 %)

Fig 3.6: VTune shows the time used by MPI for NP=32.

4. OpenMP MODEL ON THE HOST

4.1 EBI and ROS3 speedup versus EPA

An OpenMP modification was implemented in the standard CMAQ version of the CTM (CHEM) procedure since the dominant amount of time is expended there for ROS3 and GEAR solvers (see Fig. 3.4).

Fig. 4.1 shows timing of the OpenMP parallel region in the CTM for the 288 calls to CHEM in the EBI version for a 24 hour simulation. With 8 threads the speedup over the standard EPA version ranges from 1.5 to 2.3. Using an average of 1.9, this suggests that since EBI expends ~28% of the wall clock time (see Fig. 3.4) the best overall speedup to be expected is ~14%, and in practice ~11% is observed. The AERO science process dominates the wall clock time in the EBI algorithm

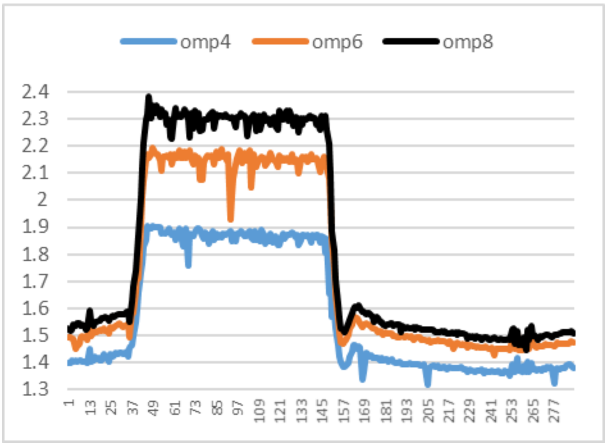


Fig 4.1: Speedup over EPA in 288 calls to CHEM with EBI for 4, 6, and 8 threads, for NP=1 MPI processes.

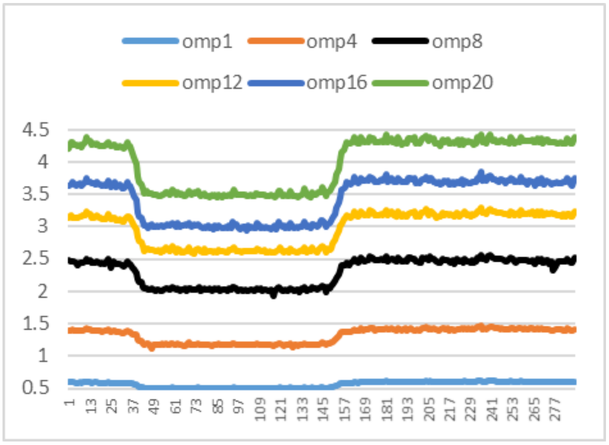


Fig 4.2: Parallel thread speedup over EPA in 288 calls to CHEM with ROS3 for 1 to 20 threads on the host, for NP=1 MPI processes.

Fig. 4.2 shows timing of the OpenMP parallel region in the CTM for the 288 calls to CHEM in the ROS3 version on the host. Since a larger fraction of the total time is expended in calls to CHEM in the ROS3 algorithm (see Fig. 3.4), the OpenMP speedup over the standard EPA version is greater than in the EBI algorithm. The average speedup over the standard EPA version (in 288 calls to CHEM) ranges from 1.3 (4 threads) to 4 (20 threads). For a 24 hour simulation, Fig. 4.3 shows that the thread speedup (OMP) versus the EPA (EPA) version in overall wall clock time is 1.4 with 20 threads. When compared to the EPA version this translates to a 29% reduction of wall clock time with 20 threads, and 20% is achieved with 8 threads. The diminution of performance gain with higher thread counts is due to the smaller partitions of work per thread calculated from 5040 blocks of cells divided amongst the number of available threads. Grid cells are partitioned into blocks of size 50 and these blocks are distributed to threads in a thread team in the OpenMP version.

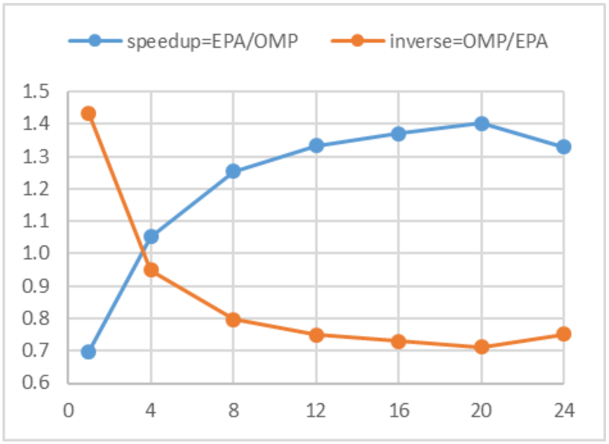


Fig 4.3: Parallel thread speedup (upper curve) of the entire 24 hour simulation over the EPA version of ROS3 for 1 to 24 threads on the host with NP=1 MPI process.

5. OpenMP MODEL ON THE Phi

5.1 ROS3 speedup versus EPA

Fig. 5.1 shows timing of the OpenMP parallel region in the CTM for the 288 calls to CHEM in the ROS3 version on the Intel Phi processor.

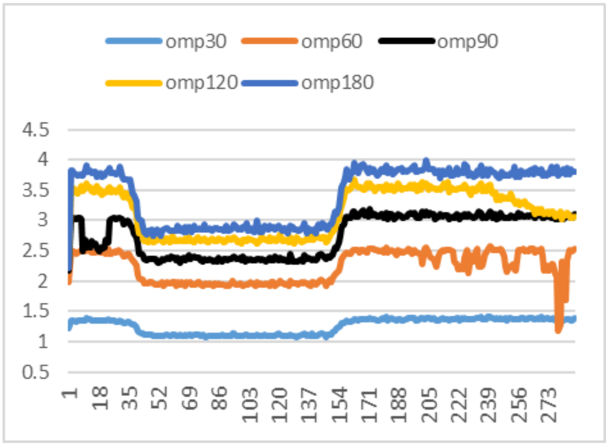


Fig 5.1: Parallel thread speedup over EPA in 288 calls to CHEM with ROS3 for 30 to 180 threads on the Phi, for NP=1 MPI processes.

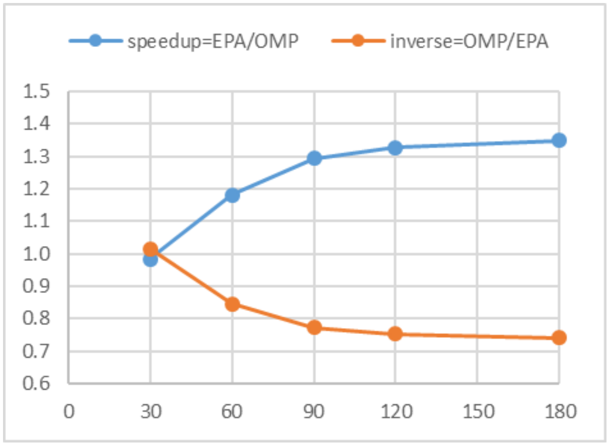


Fig 5.2: Parallel thread speedup (upper curve) of the entire 24 hour simulation over the EPA version of ROS3 for 30 to 180 threads on the Phi with NP=1 MPI process.

Fig. 5.2 shows that the thread speedup (OMP) versus the EPA (EPA) version in overall wall clock time is 1.35 with 180 threads. When compared to the EPA version this translates to a 26% reduction of wall clock time with 180 threads, and > 20% achieved with 90 threads.

5.2 Performance snapshots on the Phi

Real time performance is observable on the Intel Phi. Fig 5.3 shows the time series with the peaks representing two offload regions from the host to one Phi card. This shows utilization at 75% at the offload segment because only 180 of a possible 240 threads are active.

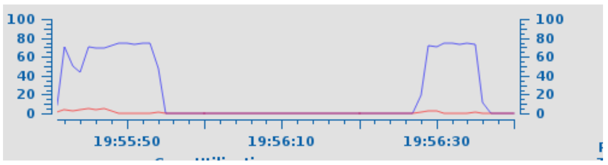


Fig 5.3: Time series snapshot with 180 threads.

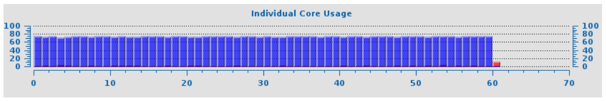


Fig 5.4: Core utilization snapshot with 180 threads.

For the same case, Fig. 5.4 shows that 60 cores are active during an offload segment, with three threads on each core making a total of 180 threads. Each call to the CTM has two offloads to the Phi corresponding respectively to the block reo-ordering and solve steps. Data is moved from CPU to the Phi with each offload: 269 MB for the reorder step, and 416 MB for the solve step.

6. NUMERICAL AND CODE ISSUES

6.1 Norms in the chemistry solver

To understand numerical accuracy metrics are used to show precision after the decomposition and solve steps of the sparse linear system Ax = y. Such metrics are easily monitored in the FSparse algorithm with an option to calculate several types of norms including |A|, |x|, and |Ax-y|, as summarized in Table 6.1. The length of the vector (Ax-y, or x) is the number of species. The "inf" norm selects the maximum value of each vector, Ax-y (residual), or x (solution), respectively. The statistic of Table 6.1 is then computed as either the mean over all 50 cells in a block, or sampled for a specific cell number in the block.

Table 6.1. Metric for chemistry solver of CMAQ 4.7.1 with ROS3 for each block of the entire domain.

value	metric	
	norm	Statistic (calculated over all cells in a block)
Residual	norm(Ax – y, inf)	mean, standard deviation, coefficient of variation.
Solution	norm(x, inf)	mean

In the FSparse method the residual remains negligibly small in the FSparse algorithm for the chemistry solver.

6.2 Comparing concentration values

Any discrepancy between predictions of the JSparse [3] and FSparse [1] algorithms in the two methods is explained by the way precision is treated in each. The Chemistry solver uses double precision arithmetic but accepts some input data from single precision variables (temperature, pressure, photolysis rates, reaction rates, etc.). Any loss in precision is amplified as the solution progresses in the three Rosenbrock solve stages. Therefore all expressions in FSparse are performed in double precision. The acid test is to compare the computed concentration values for selected species as predicted by the EPA (using JSparse) and the thread parallel version (using FSparse). Careful inspection of these concentration values for selected species (O3, NOx, etc) for the entire domain and all 24 time steps showed differences within the RMS tolerance controlling convergence in the ROS3 algorithm.

6.3 Code issues uncovered in this study

Several code issues were uncovered during this implementation using the CMAQ 5.1 download of the original EPA version from the CMAS site. These included the following:

- In `~ /models/JPROC/jproc_table/` the files `srband.f` and `intavg.f` have interface errors.
- In `bldit.cctm` script the preprocessor uses a reserved name in "`set PAR=( -Dparallel )`" that corrupts OpenMP directives.
- More than 60,000 compiler warnings such as "`This name has not been given an explicit type`"
- Warning messages of the type "`Global name too long`"
- Warning messages of the type "`Source line truncated`"
- Warning messages of the type "`A dummy argument with an explicit INTENT(OUT) declaration is not given an explicit value`"

To avoid termination of the compilation the warnings were disabled with the compiler option choice "`-warn all,nodeclarations,nounused`" while others required source code and script modifications. It was also found that higher optimization such as -O2 and –O3 produced executables that resulted in segfaults. These could be related to the choice of other compiler options, the version of the compiler used, and the platform.

7. LESSONS LEARNED

7.1 Benefits of the FSparse method

Comparing runtime performance for CMAQ 5.1 in the new OpenMP parallel version with the U.S. EPA release showed benefits such as:

- A speedup ~1.4 for the ROS3 algorithm on either multi-core host or many-core MIC processors thereby bring the wall clock time comparable to that of the EBI version (cf Table 3.2).
- A single source code version of the CTM.
- Thread parallel efficiency that was comparable to that of MPI processing.
- Hybrid MPI+OpenMP algorithms that offer more on-node compute intensity as the number of available threads rises to 100's and beyond.
- Numerical values of predicted concentration that are within the error tolerance inherent in the algorithms.

7.2 Next steps

A continuation of this work would include:

- Removal of blocking of cells and application of the CTM to individual cells to increase the accuracy of the model.
- Extension of the thread based model to the Gear solver for CMAQ.
- Exploring a relaxation of the chemistry time step convergence error criterion to further reduce runtime.
- Implementing further compiler and implementation tuning opportunities.
- A port to the 2nd generation Intel Phi processor

8. CONCLUSIONS

This report has described an analysis of CMAQ 5.1 behavior in the standard U.S. EPA release and a new thread parallel version of CMAQ for the Rosenbrock solver. Opportunities exist for speedup with an increasing number of parallel threads that reaches the range 1.4 over the standard CMAQ release for the Rosenbrock solver.

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. The new FSparse version of ROS3 offers layers of parallelism not available in the standard U.S. EPA release and is portable across multi- and many-core hardware and compilers that support thread parallelism.

REFERENCES

[1] Delic, G., 2012, 2013: see presentation at the Annual CMAS meetings (http://www.cmasecenter.org).

[2] INTEL: Intel Corporation, http://www.intel.com

[3] Jacobson, M. and Turco, R.P., (1994), Atmos. Environ. 28, 273-284