DEVELOPING CMAQ FOR MANY-CORE AND GPGPU PROCESSORS

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

1. INTRODUCTION

Previous presentations by HiPERiSM Consulting, LLC, in this conference series have reported performance results for both serial and multithread parallel versions of CMAQ (Delic, 2003-2009). This report presents results of porting the multithreaded version of CMAQ to recent multi-core and many-core processors. Examples of the former are traditional x86_64 processors while examples of the latter are General Purpose Graphical Processing Units (GPGPU).

Both Rosenbrock (ROS3) and Gear (GEAR) chemistry solver versions of CMAQ offer potential for thread parallel code development, whereas the Euler-Backward (EBI) solver does not. Recently a thread-parallel version of the CMAQ Rosenbrock solver (hereafter ROS3-HC), developed by HiPERiSM Consulting, LLC, (Delic, 2009), was delivered under contract to the U.S. EPA (See the Acknowledgements). Selected results from this study are presented here, together with an exploration of GPGPU architectures. A case study uses the thread-safe version of the CMAQ Rosenbrock solver reported on in the previous year's CMAS conference (Delic, 2009). Although some results for CMAQ 4.7 are included, this report will focus on experiences with CMAQ 4.6.1 for ease of comparison with the previous report.

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). These new parallel layers in CMAQ are suitable candidates for both multi-core and many-core targets.

2. CHOICE OF PLATFORMS

2.1 Hardware

The hardware systems chosen were the platforms at HiPERiSM Consulting, LLC, shown in Table 2.1. The GPGPU device shown in Table 2.1 was the first release with native double precision capability and 4 Gigabytes of memory. It is currently installed on the quad-core 1 (QC-1) platform at HiPERiSM. The quad-core 2 (QC-2) platform has scope for the addition of two (more recent) GPGPU devices that offer 448 cores and up to 6 Gigabytes of memory on each. Each of the two platforms, QC-1 and QC-2, have a total of 8 cores and, when combined form a heterogeneous cluster. This cluster is used for either MPI only, or hybrid thread-parallel plus MPI execution, and results for both modes are reported below.

Table 2.1. Processors at HiPERiSM Consulting, LLC

Platform	SGI Altix	quad-core 1	quad-core 2	GPGPU
Processor	Intel™ IA64 (107W)	Intel™ IA32 (X5450)	Intel™ IA32 (W5590)	Nvidia™ (C1060)
Cores per processor	1 core	4 cores	4 cores	240 cores
Clock	1.5GHz	3.0GHz	3.33GHz	1.3GHz
Band-width	6.4 GB/sec	10.6 GB/sec	64.0 GB/sec(1)	102 GB/sec
Bus speed	400 MHz	1333 MHz	1333 MHz(2)	800 MHz(4)
L1 cache	32KB	64 KB	64 KB	NA
L2 cache	1 MB	12MB(3)	256MB	NA
L3 cache	4MB	NA	8MB	NA

(1) Theoretical maximum.
(2) Value for one DDR3 DIMM per each of three channels per processor (This value drops with more DIMMs per channel).
(3) Intel's first generation of Quadcore CPUs shared L2 cache between cores.
(4) This is the on-device memory speed. Communication between the GPGPU device and the host is via a PCI Express 2.0 x 16 system interface.

2.2 Hardware bandwidth for MPI

Fig. 2.1 shows a comparison of MPI bandwidth measurements for the SGI Altix and the quad-core cluster (dual 4 core CPUs on two nodes) with NumaLink® and Infiniband®SDR interconnect fabrics, respectively. The SGI Altix is limited to 8 single core CPUs, and the quad core cluster has two nodes with 2 quad core CPUs each. The "local" curve of the quad-core cluster result corresponds to scheduling MPI processes on the same (master) node. However, the "non-local" results correspond to MPI processes distributed between the two nodes in the quad-core cluster (with the exception of two MPI processes when both processes resided on the master node). The remarkable feature of Fig. 2.1 is how the on-node

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

bandwidth tracks closely the NumaLink® results for the SGI Altix. This reflects the bandwidth boost of the quad-core architecture over previous IA32 CPU generations. Clearly, it is most beneficial for parallel execution on current multi-core cluster nodes to remain on-node as much as possible to utilize the on-node bandwidth.

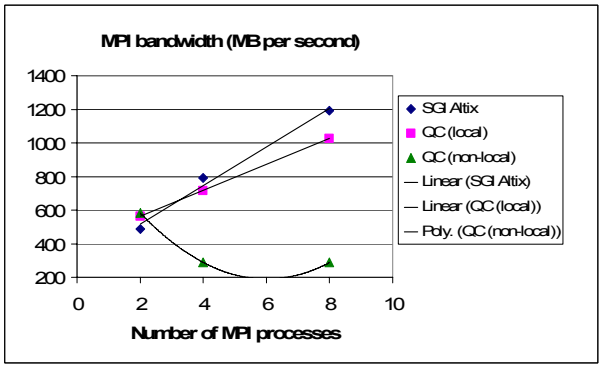

Fig 2.1: MPI bandwidth at HiPERiSM Consulting, LLC.

2.3 Compilers

Two popular compilers were used for this CMAQ study. A comparison was made for both the Intel™ 11.x and Portland 10.x Fortran compilers on 64-bit SUSE Linux operating systems. The ROS3-HC multi-threaded parallel version was compiled and executed with both compilers on all platforms shown in Table 2.1. However, while the Intel™ compiler may be implemented with the CUDA™ GPGPU programming environment (CUDA), this was deemed too labor intensive when compared with the GPGPU interface of the PGI Accelerator™ Fortran compiler (PGI). This feature-rich compiler simplifies prototype development and testing.

For each compiler four groups of optimization switches were selected corresponding to no optimization (ifc1, pgf1), some optimization (ifc2, pgf2), best optimization (ifc4, pgf4), and enabling control of arithmetic precision (ifc3, pgf3). The last choice increases execution time but constrains arithmetic operations to improve numerical precision by several orders of magnitude with either compiler. The reason for such constraints is that some CMAQ species are more sensitive to numerical differences than others, largely based on the variability in concentration magnitudes (those with largest variation being more at risk). This study found that although the highest optimization level, ifc4 of the Intel™ compiler produce the shortest runtime, in some cases it also introduces numerical differences that compromise numerical precision for a small (10%) subset of the species concentration value population. This observation concerning the Intel™ compiler applies for both Itanium2™ and current generation quad-core processors.

3. EPISODES STUDIED

For CMAQ 4.6.1 results the model episode selected was for August 14, 2006 (hereafter 20060814). This used the CB05 mechanism with Chlorine extensions and the Aero 4 version for PM modeling. For CMAQ 4.7.1 the model used the episode for August 09, 2006 (hereafter 20060809) with data provided by the U.S. EPA. 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.

4. SERIAL AND MPI RESULTS

4.1 Intel™ compiler on three platforms

Runtime results for CMAQ 4.6.1 with the three solver versions is shown in Table 4.1 for three generations of Intel platforms with the highest optimization level (ifc4) for the Intel™ compiler.

Table 4.1. Wall clock times in hours for solvers in the serial version of CMAQ 4.6.1 for the Intel™ compiler with the fastest optimization (compiler group ifc4).

CMAQ Solver	Time in hours by platform for Intel™ compiler group ifc4		
Platform	Itanium2™ ifc4	QC-1 ifc4	QC-2 ifc4
EBI-EPA	46.4	16.2	12.6
ROS3-EPA	54.2	25.7	17.3
GEAR-EPA	81.8	37.1	29.7

All three solver versions of CMAQ have gained from the evolution of commodity computer architectures with an average speed-up versus the Itanium2™ of 2.4 (QC-1) and 3.2 (QC-2). However, the speed-up of CMAQ on QC-2 versus QC-1 is in the range 1.3 – 1.5 which is only half of the potential speed-up possible between two generations of quad-core processor technology.

4.2 Intel™ versus Portland™ compilers

Typical runtime results for the standard U.S. EPA distribution of CMAQ 4.6.1 are shown in Tables 4.2 and 4.3, for Intel™ and Portland™ compilers, respectively. In both cases the "*" indicates dedicated runs and all others are for concurrent execution. Table 4.4 show the ratios of

times in corresponding cells of the preceding two tables.

Table 4.2. Wall clock times in hours for three solvers in the serial version of CMAQ 4.6.1 on the QC-1 platform for the Intel™ compiler switch groups ifc1 to ifc4.

CMAQ Solver	Time in hours by compiler group			
	ifc1*	ifc2	ifc3*	ifc4*
EBI-EPA	74.4	16.3	20.5	16.2
ROS3-EPA	147.4	25.8	30.0	25.7
GEAR-EPA	183.7	37.1	41.4	37.1

The speed-up over the Itanium2™ platform with the Intel™ compiler is in the range 2.1 to 2.9 on the QC-1 platform, depending on the solver and compiler group used.

Table 4.3. Wall clock times in hours for three solvers in the serial version of CMAQ 4.6.1 on the QC-1 platform for the Portland compiler switch groups pgf1 to pgf4.

CMAQ Solver	Time in hours by compiler group			
	pgf1	pgf2	pgf3*	pgf4
EBI-EPA	38.3	19.1	19.7	18.3
ROS3-EPA	75.8	28.5	28.5	27.5
GEAR-EPA	120.0	43.8	43.5	42.7

In the QC-1 case the difference in times for the ifc4 and pgf4 cases is due in part to the fact that the pgf4 runs were concurrent (overlapping) and this may expand wall clock time by the order of 10%.

Table 4.4. Ratios of wall clock times for three solvers in the serial version of CMAQ 4.6.1 on the QC-1 platform. The ratios are for Intel™ (ifc) versus Portland (pgf) compilers for each compiler switch group.

CMAQ Solver	Ratios for wall clock time on the QC-1 platform and compiler group			
	ifc1 / pgf1	ifc2 / pgf2	ifc3 / pgf3	ifc4 / pgf4
EBI-EPA	1.94	0.85	1.04	0.88
ROS3-EPA	1.94	0.91	1.05	0.93
GEAR-EPA	1.53	0.85	0.95	0.87

Note that, from Table 4.2, the increase in runtime for use of the Intel™ compiler group ifc3 versus ifc4 is in the range 10% to 27%, whereas for the Portland compiler the corresponding increase is in the range 2% to 8% (Table 4.3). As a result the comparative times for use of groups ifc3 and pgf3 in the respective compilers shrinks to the order of 5%. The use of the ifc3 and pgf3 compiler groups is recommended for reasons of improved precision in concentration values for some species.

4.3 MPI results

The preceding tables showed results for the standard U.S. EPA distribution with no parallel execution enabled. This section presents MPI results for EBI and Rosenbrock (ROS3) chemistry solver versions of CMAQ 4.6.1. Table 4.5 summarizes the CMAQ 4.6.1 runtimes (in hours) with the Portland compiler in an MPI implementation. Also shown there is the scaling with increasing MPI process count and it is notable that speedup departs significantly from linearity with more than 4 MPI processes.

Table 4.5. Wall clock times (in hours) and parallel scaling, for two solvers in the MPI implementation of EPA's standard release of CMAQ 4.6.1 on the HiPERiSM QC Cluster platform for the Portland compiler group pgf3.

Col x Row = NP	Time in hours (EPA)		MPI speedup versus NP=1	
	EBI	ROS3	EBI	ROS3
1 x 1 = 1	19.6	29.0	1.0	1.0
1 x 2 = 2	10.9	15.1	1.8	1.9
2 x 2 = 4	6.4	8.2	3.1	3.5
2 x 4 = 8	3.9	5.1	5.1	5.7
2 x 8 = 16	2.6	3.3	7.5	8.7
4 x 4 = 16	2.9	3.4	6.8	8.4

Corresponding to the previous table, Fig. 4.1 summarizes the CMAQ 4.6.1 MPI parallel efficiency with increasing process count. It is clear that EBI and ROS3 solvers show a steep decline in MPI parallel efficiency when NP>4. The asymptote of parallel efficiency is of the order of 50% for 16 MPI processes where CPUs are idle for half of the wall clock time (on average).

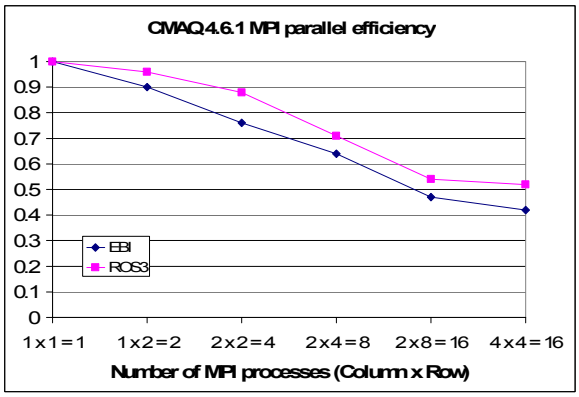

Fig 4.1: MPI parallel efficiency for CMAQ4.6.1 EBI and ROS3 solvers.

5. OpenMP PARALLEL RESULTS

5.1 Intel™ compiler results for CMAQ 4.6.1

Runtime results for the thread parallel version of CMAQ 4.6.1 with the fastest Intel™ compiler switch group (ifc4) are presented here for the parameter choices BLKSIZE=480 and

NCMAX=60. For a discussion of these parameters see the previous report in this series (Delic, 2009). Table 5.1 repeats the results from the previous year's presentation with a correction in terminology. In this table two performance metrics are introduced to assess thread parallel performance:

(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 in the ROS3-HC modified code.

Table 5.1. For CMAQ 4.6.1, with the 20060814 episode, this shows the wall clock time (in hours), speedup, and scaling of the modified version (ROS3-HC) as a function of increasing thread count on the QC-1 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	1.30	1.45
ROS3-HC (scaling)	1.00	1.41	1.51	1.68

5.2 Intel™ compiler results for CMAQ 4.7.1

Runtime results for a pre-release version of CMAQ 4.7.1 with the fastest Intel™ compiler switch group (ifc4) are presented here for the parameter choices BLKSIZE=480 and NCMAX=60. These are shown in Tables 5.2 and 5.3, for QC-1 and QC-2, respectively, for the thread parallel version.

Table 5.2. For CMAQ 4.7.1, with the 20060809 episode, this shows the wall clock time (in hours), speedup, and scaling of the modified version (ROS3-HC) as a function of increasing thread count on the QC-1 platform.

CMAQ version for Rosenbrock solver	Number of threads			
	1	2	4	8
U.S. EPA (hours)	33.0	NA	NA	NA
ROS3-HC (hours)	36.0	29.7	26.1	23.9
ROS3-HC (speedup)	0.92	1.11	1.26	1.38
ROS3-HC (scaling)	1.00	1.21	1.38	1.51

Table 5.3. For CMAQ 4.7.1, with the 20060809 episode, this shows the wall clock time (in hours), speedup, and scaling of the modified version (ROS3-HC) as a function of increasing thread count on the QC-2 platform.

CMAQ version for Rosenbrock solver	Number of threads			
	1	2	4	8
U.S. EPA (hours)	23.06	NA	NA	NA
ROS3-HC (hours)	26.06	21.7	18.6	17.3
ROS3-HC (speedup)	0.88	1.06	1.24	1.34
ROS3-HC (scaling)	1.00	1.20	1.40	1.51

5.3 Analysis of OpenMP results

The principal results in comparisons of the above tables are as follows.

- The modified version of the ROS3-HC solver for CMAQ showed typical speedup with 8 parallel threads in the range 1.3 to 1.5 over the U.S. EPA version.
- The speedup metric shows CMAQ 4.7.1, when compared to 4.6.1, has less gain in performance with increasing thread count.
- CMAQ 4.7.1 requires considerably longer runtimes compared to CMAQ 4.6.1.
- The gain in moving the U.S. EPA version from QC-1 to QC-2 is 1.3.
- The gain in moving the ROS3-HC version from QC-1 to QC-2 is 1.22 to 1.43 (depending on the number of threads).

The last two results are a consequence of CMAQ 4.7.1 shifting the balance of arithmetic operations further toward scalar work (i.e. less vector-capable work) compared to CMAQ 4.6.1. In other words, less time is spent in the chemistry solver part relative to the rest of the model.

6. HYBRID OpenMP+MPI RESULTS

6.1 CMAQ 4.6.1 runtime

Runtime results with the Portland compiler switch group pgf3 are presented here for the parameter choices BLKSIZE=240 and NCMAX=30. Table 6.1 summarizes the CMAQ 4.6.1 results for episode 20060814 with the Portland 10.3 compiler.

Table 6.1. Wall clock times (in hours) in the hybrid MPI+OpenMP version of the CMAQ 4.6.1 ROS3-HC solver on the HiPERiSM QC Cluster platform for the Portland compiler group pgf3.

Col x Row = NP	ROS3-EPA (hours)	ROS3-HC			
		Time in hours by thread count			
		1	2	4	8
1 x 1 = 1	29.0	34.8			24.1
1 x 2 = 2	15.1		15.2	13.3	12.6
2 x 2 = 4	8.2		8.0	7.5	
2 x 4 = 8	5.1		5.0		

Execution times of the standard EPA release are in the column labeled ROS3-EPA. Columns under the label ROS3-HC show results of the hybrid MPI+OpenMP modified CMAQ version with the Rosenbrock solver. The rows correspond to the MPI process count (NP) and 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.

6.2 CMAQ 4.6.1 speedup

For the hybrid MPI+OpenMP modified CMAQ version with the Rosenbrock solver, Table 6.2 shows the speedup metric corresponding to the runtimes in Table 6.1.

Table 6.2. Speedup in the hybrid MPI+OpenMP version of the CMAQ 4.6.1 ROS3-HC solver on the HiPERiSM QC Cluster platform for the Portland compiler group pgf3.

Col x Row = NP	ROS3-HC vs ROS3-EPA Speedup by thread count			
	1	2	4	8
1 x 1 = 1	0.83			1.20
1 x 2 = 2		1.00	1.14	1.20
2 x 2 = 4		1.03	1.10	
2 x 4 = 8		1.00		

7. OPPORTUNITIES FOR PERFORMANCE

7.1 Multi-thread CMAQ on a CPU

For ROS3-HC successful performance relies on enhanced vector loop capability in the Rosenbrock solver to take advantage of instruction level parallelism. As noted previously (Delic, 2009), all 43 candidate loops in the OpenMP version of CMAQ's Rosenbrock solver do vectorize. All that remains is to make the best choice of vector length, NCMAX, and BLKSIZE, such that BLKSIZE/NCMAX has no remainder. The choice of values for the best BLKSIZE and NCMAX combination depends on the cache and memory architecture. Empirical studies (Delic, 2009) determined that, for host CPU architectures such as QC-1 and QC-2, NCMAX should be in a range 18 to 90 and BLKSIZE ≤ 480. Typically, values of vector loop length NCMAX > 90 and larger BLKSIZE choices increase the runtime on multi-core commodity processors.

7.2 Many-thread CMAQ on a GPGPU

A successful port to a GPGPU device requires either multilayered loop nests, or very long (single) loops to reap the benefits of the many threads such devices employ. This suggest that if the NCMAX "vector" length could be made very large there could be throughput benefits for a CMAQ parallel version adapted for such devices. If at the same time, the number (BLKSIZE) of cells in a block could be increased, then the number of such blocks passed to the solver would be reduced. Consequently, the number of calls to the CMAQ chemistry solver algorithm is fewer, and loops spanning the entire block would be off-loaded to the many-core device. This motivates the interest in exploring the port of the ROS3-HC thread-safe version of CMAQ to a GPGPU device.

7.3 Benchmark of CMAQ loop nests

As a sequel to the above proposal the vector length for loops over domain cells in the CMAQ 4.6.1 ROS3-HC version was set equal to the block size, NCMAX= BLKSIZE, with values of BLKSIZE incremented in a wide range. Table 7.1 shows the vector length increments chosen for this analysis on both the host CPU (X5450) and the GPGPU device (C1060). The domain size of 2,276,640 cells corresponds to the episodes described in Section 3 above.

Table 7.1. Range of vector lengths used in benchmarks on the host CPU and GPGPU device for a 279x240x34 domain size.

Number of blocks = number of calls to the solver	Blocksize = vector length
8894	256
4447	512
2224	1024
1112	2048
556	4096
278	8192
139	16384
70	32768

For timing benchmarks several loop nests from the CMAQ ROS3-HC thread parallel version were selected. Table 7.2 lists the loop names used below in figure legends. Details of the source code constructs are described elsewhere (Delic, 2010).

Table 7.2. Loop names for loop nest benchmarks.

CMAQ-ROS3 solver loops used in benchmarks		
Loop name	Number of loop nests in parallel region	Loop nest depth
L478	1	2
L522-37	2	2
L1240	2	2 and 1
L1284	3	1, 2, and 1

The Portland 10.6 Fortran compiler was used with the pgf3 compiler group. For the host CPU the loops were compiled as single threaded loops. In the GPGPU case, the Portland Accelerator™ compiler generated kernels and data movement for the target device by recognition of accelerator directives that encapsulated loop nests in the

benchmarks. In conducting these benchmarks no additional GPGPU optimizations were performed.

8. COMPARISON OF CPU AND GPGPU

For the benchmarks listed in Table 7.2, in the ROS3-HC version of CMAQ, results of benchmarks are presented for two performance metrics as a function of increasing vector length:

- Ratio of times on the GPGPU device versus the host CPU.
- Ratio of time for GPGPU computation (kernel) versus time for moving data between host and GPGPU device (data).

The first metric indicates a gain for the GPGPU over the CPU performance when the value is less than unity. The second metric is an indicator of the level of computational intensity (flops per memory operation) in each benchmark. Results of these metrics for the range of vector length in Table 7.1 are shown in Figs. 8.1 and Fig. 8.2, respectively.

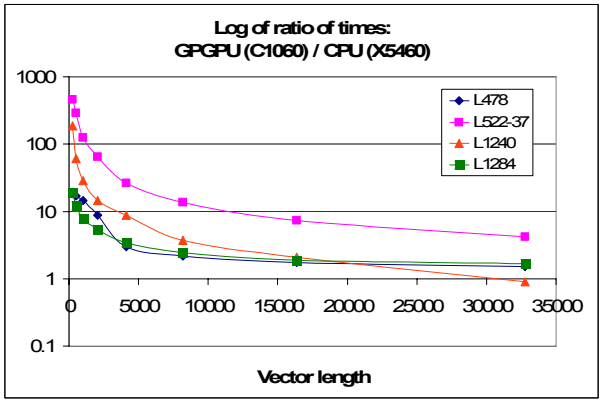

Fig 8.1: Ratio of time on the GPGPU device versus the host CPU for each of the four CMAQ benchmarks in Table 7.2.

Fig.8.1 shows that, with smaller values of the vector length, the times on the host CPU are typically orders of magnitude faster than the GPGPU device. However, as vector length increases, there is a rapid gain for GPGPU performance with the L1240 benchmark outperforming the host CPU time for the longest vector length.

However, one of the penalties for using the attached GPGPU device is the cost of moving data between the host and the device. For three of the benchmarks, Fig.8.2 shows the ratio of computation time to data movement time when the GPGPU is utilized. For the largest vector lengths this ratio is ≤ 1 suggesting that the cost of data movement dominates the kernel computation time.

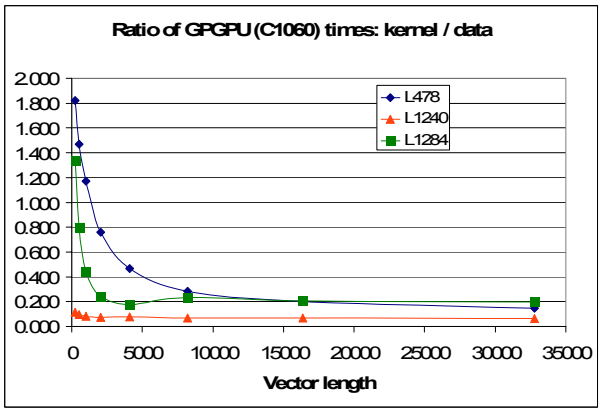

Fig 8.2: Ratio of time spent in kernels on the GPGPU device versus the time required to move data between the host CPU the GPGPU device for three CMAQ benchmarks in Table 7.2.

9. CONCLUSIONS

This report has described a successful port to recent multi-core CPUs of a parallel hybrid (OpenMP and MPI) version of CMAQ for the Rosenbrock solver. Exploratory benchmarks with selected loops on a many-core GPGPU device suggest that opportunities exist for CMAQ on such devices, but more work is needed to improve performance. Further opportunities remain for thread parallelism in other parts of the CMAQ model outside of the chemistry solver.

REFERENCES

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

Delic, 2003-2009: see presentations at the Annual CMAS meetings (http://www.cmasecenter.org).

Delic, 2009, 8th Annual CMAS Conference, Chapel Hill, NC, October 19-21, 2009.

Delic, 2010, Technical Report HCTR-2010-5 at http://www.hiperism.com

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

ACKNOWLEDGEMENTS

Part of this work was performed by HiPERiSM Consulting, LLC, as subcontractor to Computer Sciences Corporation, under U.S. EPA SES3 Contract GS-35F-4381G BPA 0775, Task Order 1522