A NEW PARALLEL VERSION OF MOVESMRG FOR SMOKE

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

1. INTRODUCTION

This presentation reports on the first major thread parallel enhancements for the SMOKE 3.1 model as developed by the U.S. EPA [CMAS]. These modifications replaced the serial version of the MOVESMRG Fortran procedure with a thread parallel version. This procedure merges emission factors with activity data to create the on-road mobile emissions input required by CMAQ. It is one of the most time-consuming components of SMOKE and therefore it is of interest to reduce the wall clock time expended there.

The new version (hereafter SMOKE-HC) offers performance improvement over the U.S. EPA serial release (hereafter SMOKE-EPA). MOVESMRG contains algorithms that assume serial processing in a time step loop (on T) in one hour increments for multiple days. This is encapsulated in a loop over reference counties and contains voluminous I/O and memory intensive processing, mixed with branching logic and arithmetic operations. In the parallel version of MOVESMRG performance limiting procedures were hoisted out of the parallel region and preprocessed in a separate serial loop on T followed by a second parallel loop on the time step. The parallel layers developed at HiPERiSM added thread-level parallelism but found negligible scope for instruction-level parallelism (at the vector loop level) in the SMOKE algorithms. The parallel version of SMOKE was tested with the Portland Group® compiler [PGI] and results are reported for multi-core platforms from the Intel Corporation® (INTEL) and Advanced Micro Devices® (AMD). The next section details the HiPERiSM test bed and is followed by a description of the parallel modifications and results.

2. TEST BED ENVIRONMENT

2.1 Hardware

The hardware systems chosen were the platforms at HiPERiSM Consulting, LLC, shown in Table 2.1. The two platforms shown in the table (INTEL and AMD) have a total of 8 and 48 cores, respectively. Each of these two cluster nodes was used separately for thread-parallel OpenMP execution as discussed 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
Total memory	128GB	96GB
(1) Per core, (2) Per socket		

The total memory shown in Table 2.1 does not represent the upper limits available with this generation of hardware but for now is configured for optimal performance.

2.2 Compilers

Compilers available for the platforms in Table 2.1 include those from Intel (12.1), Portland (12.5/13.4) and Absoft (11.5) on 64-bit Linux operating systems and hardware. However, for this report, the HiPERiSM Consulting, LLC, version of SMOKE-HC with multi-threaded parallelism was compiled and executed only with the Portland compiler. Results for other compilers will follow at a later date.

3. EPISODE STUDIED

For all SMOKE results reported here the model episode selected used data provided by the U.S. EPA [BAEK]. This episode includes several scientific models but only the rate per distance one was used throughout this report because this had the largest fraction of wall clock time expended in MOVESMRG. This model was compiled and executed for five cases with increasing number of time steps (T) in each, corresponding to T=25, 43, 73, 97, and 121, respectively. In each case the

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

number of reference counties was fixed at 146, and the number of species is fixed at 142. The number of sources depends on the county and ranges up to 447696. The number of grid cells depends on the source and varied up to ~35.

4. ANALYSIS OF MOVESMRG CODE

4.1 Background

The MOVESMRG model in SMOKE consumes the largest fraction of time in the case of the rate per distance simulation. This was used in the cases reported here and also in those that come with the SMOKE 3.1 download [CMAS]. A profile and examination of the loop structure was undertaken to localize the workload distribution within MOVESMRG. This loop structure is shown schematically in Table 4.1 and a profile showed that this is where the most execution time is expended within MOVESMRG.

Table 4.1. Fortran nested loop structure in movesmrg.

TARGET	LOOP INDEX
inventory counties	DO I = 1, NREFC … [I/O PERFORMED]
time steps	DO T = 1, NSTEPS
sources in inventory county	DO S = 1, NREFSRCS(I)
grid cells for source	DO NG = 1, NSRCCELLS(SRC)
pollutant-species combos	DO V = 1, NSMATV
pollutant-species combos	END DO
grid cells for source	END DO
sources in inventory county	END DO
time steps	END DO
inventory counties	… [I/O PERFORMED] END DO

4.2 Parallelization strategies

With a view to thread parallelism potential the nested loop structure in MOVESMRG was examined starting from the innermost loops.

The grid and species loops (NG,V) are candidates for parallel implementations, but in the current form they contain both serial code constructs and subscript references that are indirect, i.e. do not directly depend on the respective loop index NG or V. As a result vector processing is inhibited. Furthermore, inspection of the arithmetic operations shows that they are not numerous and most use variables that do not dependent on loop indexes. Also these loops contain multiple logical branches. As a result the assessment was that the cost of thread synchronization may overcome any gains in parallel performance scaling at this level. This would result in a parallel implementation at the NG or V loop level that takes longer to execute than the serial code. An added factor is that for serial code, current compiler technology would implement scalar optimizations on the DO NG and DO V loops that are more efficient than any gains from a parallel implementation. This assessment was confirmed in the parallel performance observed with a coding prototype.

At the next higher loop level the source loop (S) is a good candidate for a parallel implementation, but requires the modification and removal of serial algorithms and potential synchronization requirements in the contained loops for NG and V variables. One issue in the DO S loop is that this loop maps the S index into a source index, SRC, and NG into a CELL index. In addition species are also indirectly referenced. A second parallel prototype showed low efficiency.

Further up the loop nest chain is the time step loop on T. This loop is inherently serial in the original SMOKE-EPA version, and could be avoided as a parallelization target because the best target for parallelization is the outermost loop. However, the county loop has the problem that it includes procedure calls to perform file reads and writes with lengthy formatted I/O operations and storage into memory. While the opportunity for parallelism exists, it would require careful synchronization of threads to avoid I/O collisions. As a result the level of effort for a parallel implementation at the county loop (I) would be significantly higher than the effort involved on parallel implementation of contained loops.

The outcome of this analysis of parallel strategies, is that the time loop on T was selected as the thread parallel index. As a result, to create a parallel MOVESMRG version, significant code modifications were required to:

a) remove some contained procedure calls that invoke I/O,
b) replace inherently sequential algorithms that require serial processing, and
c) classify variables into shared and private groups in the parallel region.

The parallel implementation hoists I/O operations into a separate serial loop on T and stores required values in memory as a preprocessing step before a subsequent parallel loop on T contained in a thread parallel region. In a parallel task scheduling algorithm this loop on T performs arithmetic and logical branch operations with separate threads assigned to separate T values. Since parallel processing on the T index is in random order, results computed inside the

parallel region need to be stored to memory (in arrays indexed on T) for retrieval outside the parallel region for I/O loop operations serial in T.

4.3 I/O and memory usage

The I/O and memory usage in MOVESMRG needs some clarification. Already in the serial SMOKE-EPA version, memory usage is intensive and scales with the number of time steps T. This is the case despite some options that attempt to optimize memory usage. For SMOKE-HC the expected gain in wall clock time comes at the cost of an increase in memory demand that escalates with the number of time steps T. Table 4.2 compares the memory requirements for the serial and parallel versions of SMOKE. The increase in memory usage for the HiPERiSM parallel version is not a major sacrifice since current market cost of memory is approximately $8.50 per gigabyte (i.e. the 96GB on the INTEL platform cost $816). This represents 26% of the cost per gigabyte approximately 4 years earlier.

Table 4.2. Memory usage in MOVESMRG

Case and number of time steps T (hours)	Memory usage by MOVESMRG version (Gigabytes)	
	EPA	HiPERiSM
25	3.0	6.7
49	3.6	10
73	4.3	14
97	4.9	18
121	5.5	22

4.4 Status of parallel code

The thread parallel strategy described above can only succeed if there is sufficient coarse grain parallel work for each thread. However, because of the diminishing speed up observed at higher thread counts, it is clear that the amount of work per thread is reducing. In addition, not all counties have the same amount of work because of widely differing ranges of the innermost loops in Table 4.1. The example of Fig. 4.1 shows that only 28% of the 146 counties experience a parallel speed up in the range 2 to 5.5 with 4 or more threads. It is enhanced parallel performance in these counties that contributes to the over-all speed up of the entire simulation.

These issues continue to be investigated and results presented here are preliminary, but sufficient to demonstrate proof-of concept.

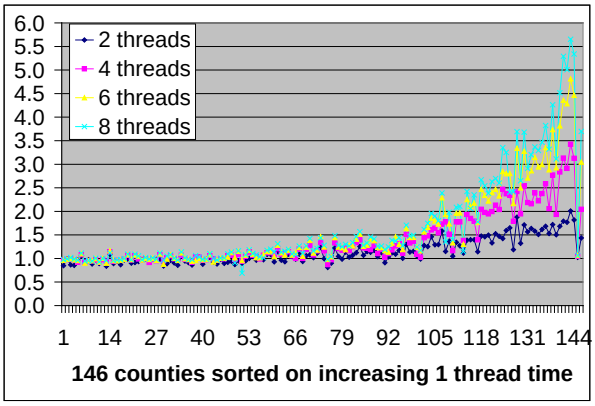

Fig 4.1: Results of a test case (T=97) with the parallel version of SMOKE-HC on the INTEL node for the number of threads in the legend. This shows parallel thread scaling for each of 146 counties sorted (left to right) on increasing time for one thread.

5. PARALLEL PERFORMANCE RESULTS

5.1 Test case runs

Two performance metrics are introduced to assess thread parallel performance in the SMOKE-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 five cases of Table 4.2, and their corresponding time step count T, Table 5.1 shows the runtimes in hours for the serial version of SMOKE and the corresponding parallel runtimes for increments in the thread count. The number of cores on the INTEL platform of Table 2.1 is limited to 8, but that of the AMD platform is considerably larger. Platform comparison shows an EPA serial version wall clock time smaller on INTEL than on the AMD node. Noteworthy is the 2.5%-7% speed up of the single thread parallel version over the serial EPA code. As the thread count increases for the parallel version, the successive improvement in wall clock time diminishes. Of special significance is the scaling result for the T=121 case. On the AMD platform, with 24 threads, the speedup is 3.2 times faster than the serial (SMOKE-EPA) time.

For each of the five cases of Table 5.1, Fig. 5.1 and Fig. 5.2, show wall clock time results on INTEL and AMD nodes for the respective number of threads listed. Correspondingly, Table 5.2, Fig. 5.3 and Fig. 5.4, show the thread scaling results for parallel performance.

Table 5.1. Portland compiler wall clock times (in hours) for serial U.S. EPA (SMOKE-EPA) and parallel (SMOKE-HC) versions of SMOKE on two platforms.

Serial and parallel thread count	Platform	Case and number of time steps T				
		25	49	73	97	121
serial	INTEL	1.21	1.71	2.16	2.65	3.14
1		1.18	1.59	2.03	2.48	2.95
2		1.04	1.28	1.52	1.75	2.00
4		1.00	1.12	1.30	1.45	1.60
6		0.95	1.05	1.18	1.32	1.41
8		0.91	1.02	1.12	1.23	1.34
serial	AMD	2.36	3.42	4.46	5.39	6.40
1		2.27	3.20	4.16	5.07	5.97
2		1.82	2.33	2.84	3.35	3.86
4		1.64	1.96	2.24	2.55	2.86
6		1.63	1.86	2.10	2.39	2.60
8		1.61	1.79	2.03	2.29	2.55
10		1.54	1.74	1.90	2.07	2.25
12		1.50	1.68	1.83	1.99	2.16
16		1.46	1.62	1.78	1.93	2.04
20		1.50	1.63	1.78	1.90	2.17
24		1.45	1.59	1.73	1.87	1.98

Table 5.2. Thread scaling with the Portland compiler for the SMOKE-HC version of SMOKE on two platforms.

Parallel thread count	Platform	Case and number of time steps T				
		25	49	73	97	121
2	INTEL	1.13	1.24	1.34	1.42	1.47
4		1.18	1.42	1.57	1.71	1.85
6		1.23	1.52	1.73	1.88	2.09
8		1.29	1.56	1.82	2.02	2.20
2	AMD	1.24	1.37	1.46	1.51	1.55
4		1.38	1.63	1.85	1.98	2.09
6		1.39	1.72	1.98	2.12	2.30
8		1.41	1.79	2.04	2.21	2.35
10		1.47	1.84	2.19	2.45	2.66
12		1.51	1.91	2.28	2.55	2.76
16		1.55	1.98	2.34	2.63	2.93
20		1.51	1.97	2.34	2.66	2.75
24		1.56	2.01	2.41	2.72	3.02

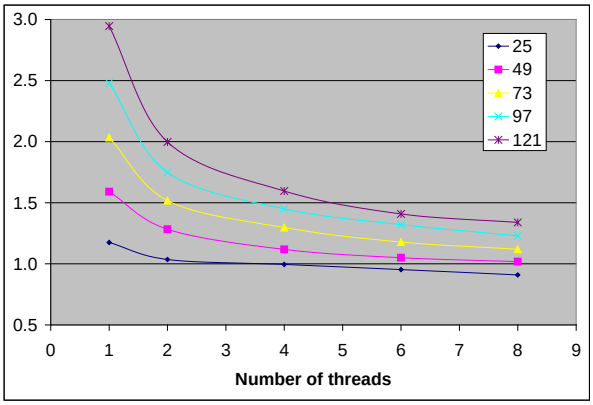

Fig 5.1: Wall clock time (hours) for five test cases with the parallel version of SMOKE-HC on the INTEL node.

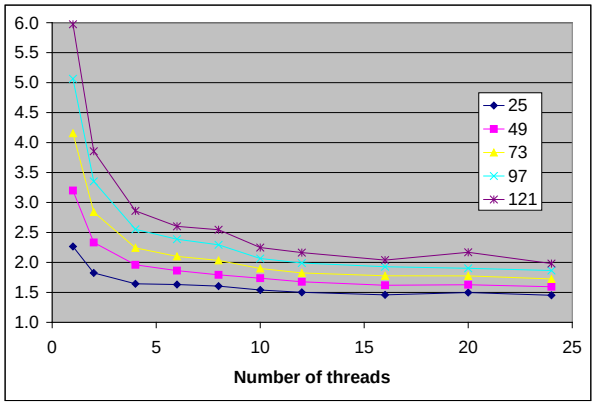

Fig 5.2: Wall clock time (hours) for five test cases with the parallel version of SMOKE-HC on the AMD node.

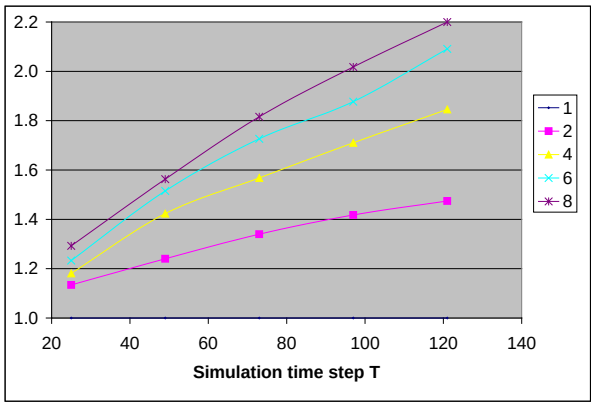

Fig 5.3: Thread scaling for five test cases with the parallel version of SMOKE-HC on the INTEL node for the thread counts in the legend. The 1 thread result is unity.

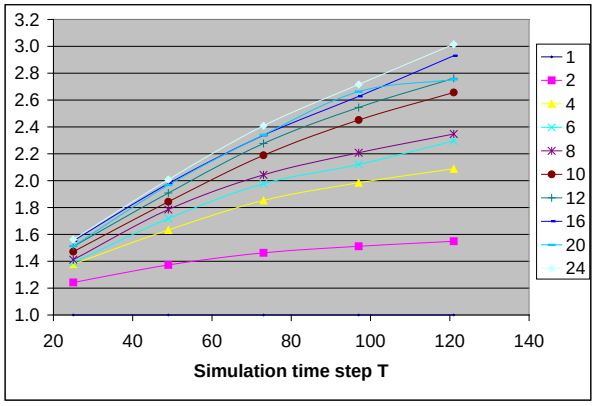

Fig 5.4: Thread scaling for five test cases with the parallel version of SMOKE-HC on the AMD node for the thread counts in the legend. The 1 thread result is unity.

5.2 Trend line extrapolations

Trends in values for the parallel times in Table 5.1, Fig. 5.1, and Fig. 5.2, may be accurately fitted with linear trend lines as a function of T. These are

then used to establish expected runtimes for much larger time step T ranges. These extrapolations are shown in Figs. 5.5 and 5.6, for INTEL and AMD platforms, respectively. The thread counts are designated in the legend.

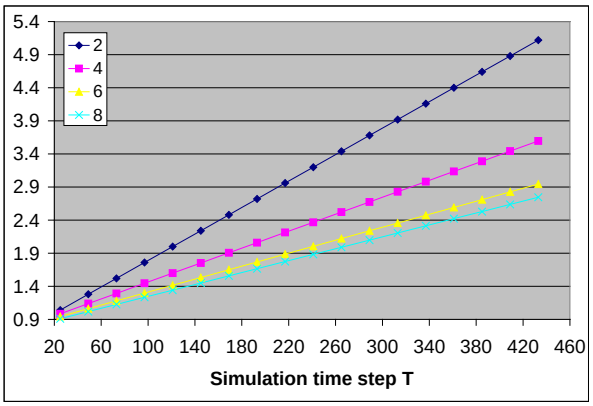

Fig 5.5: Extrapolation of Wall clock time (hours) with the parallel version of SMOKE-HC on the INTEL node.

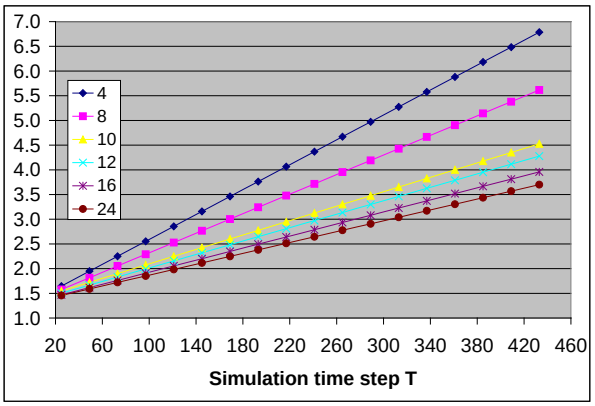

Fig 5.6 Extrapolation of Wall clock time (hours) with the parallel version of SMOKE-HC on the AMD node.

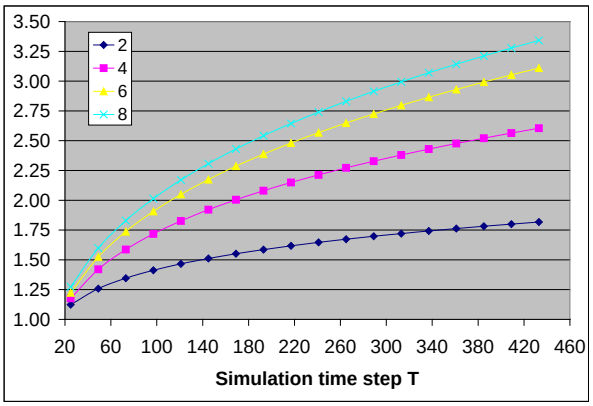

Fig 5.7: Extrapolation of thread scaling with the parallel version of SMOKE-HC on the INTEL node.

Corresponding extrapolations for thread scaling, as functions of time step count T, are shown in Figs. 5.7 and 5.8, for INTEL and AMD platforms, respectively, for the thread counts designated in the legend. Trend lines for these were obtained from fits to the results of Figs. 5.3 and 5.4 with a simple power law.

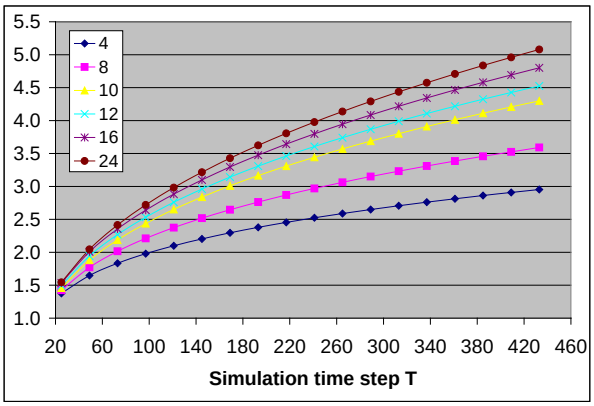

Fig 5.8 Extrapolation of thread scaling with the parallel version of SMOKE-HC on the AMD node.

6. SMOKE workload throughput

6.1 Comparing hardware platforms

There are significant differences in wall clock time and thread scaling between AMD and INTEL platforms for the same model simulation of an individual case at each time step count T. However, with more threads better performance is expected. One example of this is shown in Fig. 6.1. The wall clock time extrapolation as a function of time step (T) is for two platforms: INTEL with 4 threads and AMD with 12 threads. These used the trend line extrapolations introduced in the previous section.

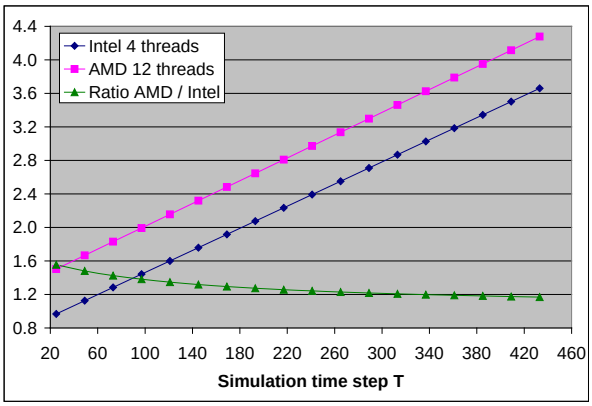

Fig 6.1: Extrapolated wall clock time (hours) versus time step (T) for the parallel version of SMOKE-HC on INTEL and AMD nodes with 4 and 12 threads, respectively. The ratio of times (AMD / INTEL) at each T is also shown.

The ratio (AMD / INTEL) of the wall clock times at each T value is also shown in Fig. 6.1 and demonstrates a reduction as the value of T increases. Actual wall clock times for cases with T in the extrapolated range (beyond 121) may differ.

6.2 Workload throughput metric

It is of interest to measure performance for workload throughput. In the case of SMOKE the workload is concurrent individual parallel runs on either platform, where each has a wide range in time steps (T). To measure performance of workload throughput a suitable metric is:

the number of simulation weeks completed per wall clock hour.

For each wall clock hour this counts how many multiples of 7 x 24 (= 168 hours) steps in T complete.

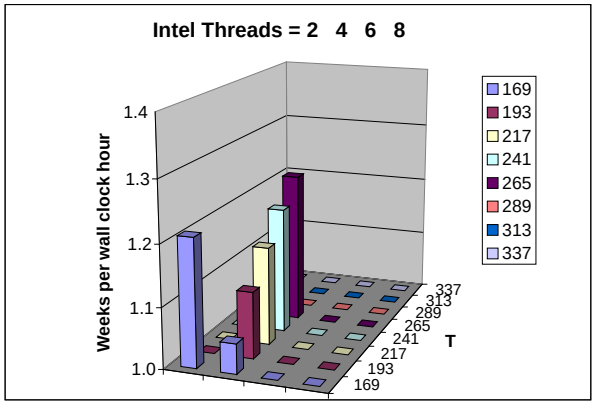

Fig 6.2: Workload throughput metric with the parallel version of SMOKE-HC on the INTEL node as a function of the thread count and number of simulation time steps.

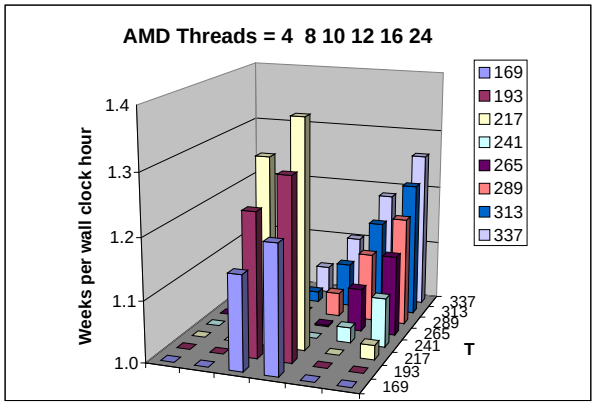

Fig 6.3: Workload throughput metric for the parallel version of SMOKE-HC on the AMD node as a function of the thread count and number of simulation time steps.

Using the extrapolations found in Section 5.2 a simple model populates either platform with as many concurrent cases allowable up to the available limits of memory and core count. These population samples are examined to see which maximize the workload throughput metric defined above. Figs. 6.2 and 6.3, respectively, present results of the workload metric for INTEL and AMD platforms. For example, on the INTEL node the best metric value of 1.2 is for 3 x 2-thread runs with T=169 each, or 1.3 for 2 x 4-thread runs at T= 265. The AMD workload equals the INTEL T=169 metric value with 3 x 12-thread runs and exceeds it at 1.37 for 3 x 12-thread runs with T=217 each. Clearly, although the INTEL platform is the speedier of the two, there is more opportunity for increased SMOKE workload throughput on the AMD node.

7. CONCLUSIONS

This analysis compared runtime of SMOKE in a new OpenMP thread parallel version with the U.S. EPA release. The results indicated that with 121 time steps the multi-threaded parallel speedup over the EPA version was 2.3 with 8 parallel threads (INTEL node), and 3.2 with 24 threads (AMD node), rerspectively. However, the AMD platform offered the best workload throughput capacity for multiple runs with longer time step ranges because of a higher core count.

Further opportunities remain for thread parallelism in other parts of the SMOKE model outside of MOVESMRG and work in this direction continues at HiPERiSM Consulting, LLC. The new version of SMOKE-HC offers layers of parallelism not available in the standard U.S. EPA release and may be ported to hardware and software that supports multiple parallel threads.

REFERENCES

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

BAEK: The author gratefully acknowledges SMOKE input data provided by B.H. Baek (UNC Institute for the Environment) and the Emissions Inventory and Analysis Group, OAQPS, U.S. EPA.

CMAS: The SMOKE model is available at
http://www.smoke-model.org

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

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