A TIMING AND SCALABILITY ANALYSIS OF THE PARALLEL PERFORMANCE OF CMAQ v4.5 ON A BEOWULF LINUX CLUSTER

Shaheen R. Tonse*
Lawrence Berkeley National Lab., Berkeley, CA, USA

1. INTRODUCTION

The goal of this activity is to improve the computational efficiency of CMAQ so that the computational demands of large modeling domains or long time periods can be met with simulations in a reasonable amount of time. When conducting time-evolving reactive flow simulations with chemistry and transport on 3 dimensional spatial grids, the computational power available usually places limits on the spatial and temporal resolution, as well as the detail to which the chemistry can be simulated. Parallel computing is one way to lower this burden. CMAQ has been released as a parallel code for several years. Parallelism in CMAQ is accomplished through the distributed parallelism method in which identical versions of the executable code run in parallel on multiple processor elements (PE) of a computer with an interconnect between PEs to transfer data and messages. The MPI (Message Passing Interface) package facilitates communications between PEs. Each PE knows its identity and its responsibilities. Generally in spatially gridded problems the grid itself is partitioned, with a different sub-domain assigned to each PE. It is the responsibility of the programmer to insert MPI subroutine calls in the code at locations where it is necessary for a PE to gather/send information about neighboring grid cells from/to another PE.

During the initialization phase of a CMAQ run the grid is partitioned among a user-specified number of PE's, assigning approximately equally sized, contiguous portions of the grid to each. The user has decides the number of divisions in the column- and row-wise directions. With each processor now responsible for calculations of a fraction of the total grid we see significant speedup over simulations that use a single processor.

In general for parallel codes, improvement in performance does not scale linearly with number of PEs. Some of the causes for this are:

1. Parts of the code are simply not parallelizable and execute redundantly on all the PEs.

2. Load imbalance between PEs causes those with lower loads to wait for the others until they have finished a task

3. Increased inter-PE communication costs (relative to actual computational costs) as the number of computational sub-domains increases or due to less-than-optimal choice of sub-domain topology.

4. Operations whose cost is dominated by startup costs (latency) and not so much by the amount of work or the reduced transfer of data. Disk accesses are in this class.

Bottlenecks to efficient parallelization can occur for any or all of these reasons. A measure of the inefficiency is the quantity scalability. E.g. if the number of PE's is increased from one to four, but the speedup is only a factor of two, then we say that the scalability is 50%. It is as if each PE has only 50% of the strength it has in a single PE, serial job. Generally scalability falls as the number of PEs increases.

When remedying the problem it is important to keep in mind the ultimate goal, which is to reduce the simulation time to an acceptable amount of time. Many users consider a 3-day simulation to be fine, with the limit of acceptability to be about 5 days. If it is possible to achieve this by simply assigning more PEs to the problem, then the user should do so. However often the computing resources are shared between users or charges are based on PE-hours utilized, and it is not possible to simply throw more resources at the problem. In a modeling project on which we are involved, (using CMAQ with a 190x190x27 grid, SAPRC99, for 124 modeling days), the time to conduct a simulation with 9 PEs is 20 days. Given that input conditions and parameters will have to be varied, multiple simulations will have to be conducted. Clearly it is desirable to use a larger number of PEs, however, because of current scalability problems we do not use more than 9.

[*] *Corresponding author:* Shaheen R. Tonse, MS 90K, Lawrence Berkeley Lab., Berkeley, CA, 94720; e-mail: tonse@lbl.gov

In this study we have conducted timing tests of parallel CMAQ v4.5 with different numbers of PEs and analysed the results, isolating the locations and causes of loss of scalability, and providing suggestions for remedying them.

2. DETAILS OF THE SIMULATION AND COMPUTING CLUSTER

The simulation is conducted using CMAQ v4.5 with the SAPRC99 gas-phase mechanism. The simulated domain is shown in Figure 1.

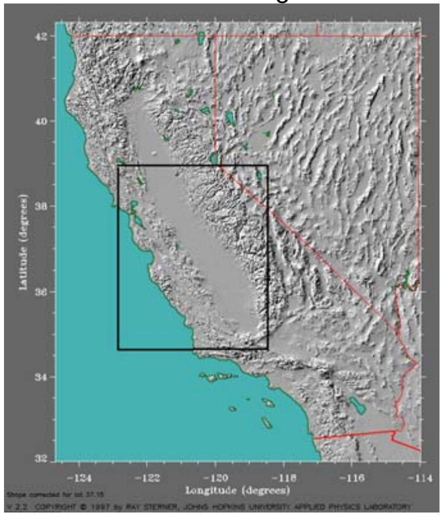

Figure 1 The domain covers the California central coast and the Central Valley, and both the San Francisco and Sacramento metropolitan areas.

The domain has 96 cells horizontally, 117 cells vertically, with each cell side being 4 km. The vertical extent of the domain is up to pressure of 100 mbars, about 16km above sea level. The meteorology is that of a 5-day episode from 29th July to 3rd Aug. 2000. The 50 original vertical layers of the MM5 simulation have been consolidated to 27, giving a total of $96\times117\times27 \approx$ 300K cells. Emissions come from Area, Point, and Biogenic and Motor Vehicle (MV) sources.

The CMAQ simulations are conducted as parallel simulations on a 28-node Linux cluster with two Athlon processors and 2 G Bytes RAM per node. The cluster is rack-based and has a fast 2 Gbits/s Myrinet Interconnect. The cluster can be viewed at (http://eetd.lbl.gov/AQ/stonse/mariah/)

Depending on the needs of the simulation we choose between the SMVGEAR (Sparse Matrix Vectorized Gear) and the EBI (Euler Backward Iterative) solvers. SMVGEAR (the stiff solver), is the more computationally expensive of the two, consuming a large share of the total CPU time. Our main reason to occasionally resort to this solver is that the Process Analysis module, a sophisticated model diagnostic, has only been implemented to work with SMVGEAR. The EBI solver, which was first released with CMAQ version 4.5, produces accurate results and has resulted in considerable speed-up compared to SMVGEAR. The computational load of the EBI module is comparable to that of the advective and diffusive transport modules. We generally use EBI for simulations that do not require Process Analysis. For both solvers, scalability is observed to decrease as the number of PEs increases. For the other physical processes: the “yamo” advection option is used, “eddy” vertical diffusion, “multiscale” horizontal diffusion, and “noop” for PinG, aerosols, and clouds.

3. METHOD

We have inserted timing calls into CMAQ to measure the elapsed time between various points in the code. The actual timing is provided by the MPI function MPI_WTIME, which returns the total clock time elapsed since the beginning of the simulation. Since we are the sole user of the cluster and the memory per node is large enough that CMAQ never undergoes memory swap, the result returned by MPI_WTIME is an accurate measure of time elapsed within the code, be it time spent in calculation, communication or disk access. We have encapsulated MPI_WTIME within a user-friendly interface. This allows simultaneous measurement of multiple elapsed times in different parts of the code. In general, most of the timing calls have been made in the scientific processes section of the code (Subroutine SCIPROC) or its daughter subroutines, which calculate the chemistry, horizontal/vertical diffusion, and horizontal/vertical advection. Within these routines, measurements are made of times spent for pure calculation, inter-PE communication, and disk access. Elapsed times are written to the CMAQ log files at user-defined intervals. In the post-processing phase the information is gathered and processed for visualization by a simple C++ programme. We use quantities like Scalability (also called Parallel Efficiency) and Load Imbalance to characterize the parallel performance. Scalability is defined as:

$$(\text{Time}_{\text{serial}}) / (\text{Time}_{\text{parallel}} \times \text{No.PEs})$$

Load imbalance between several PEs which have been timed for a particular task is defined as:

$$(\text{Time}_{\text{slowest}} - \text{Time}_{\text{mean}}) / \text{Time}_{\text{mean}}$$

4. RESULTS AND ANALYSIS

Timing results have been made for 1 PE serial and several parallel cases. The parallel cases generally use 9 and 18 PEs, in which the grid is split 3×3 and 6×3 respectively in the column- and row-wise directions. A few other grid-partitioning schemes have also been tried. All results use identical inputs and identical scientific processes (with the exception of the chemical solver) and are for the same 24-hour simulation period.

4.1 Serial Simulations

Tables 1 and 2 show times spent in various scientific modules for serial SMVGEAR and EBI simulations, respectively.

Table 1 Times for a serial SMVGEAR solver simulation, showing time spent in chemistry (CHEM), horizontal advection (HADV), horizontal diffusion (HDIFF), vertical diffusion (VDIF), vertical advection (ZADV) and their sum. The total simulation time was 255000 s. "Total time" column is the total time spent in a particular scientific module. Fraction is the fraction of time for that module divided by the sum.

Module name	Total time (s)	Fraction
CHEM	238000	0.94
HADV	7010	0.027
HDIFF	634	-
VDIF	6680	0.027
ZADV	733	-
Sum	253000	≡ 1

Table 2 Times for a serial EBI solver simulation. The total simulation time was 24920 s.

Module name	Total time (s)	Fraction
CHEM	7370	0.32
HADV	7610	0.33
HDIFF	633	0.027
VDIF	6682	0.29
ZADV	733	0.03
Sum	23028	≡ 1

It is apparent that chemistry dominates in SMVGEAR, whereas for EBI, the chemistry (CHEM), horizontal advection (HADV) and vertical diffusion (VDIF) all use comparable amounts of time. Vertical advection (ZADV) and horizontal diffusion (HDIFF) use negligible amounts, however we did continue to monitor them for scalability in parallel simulations in the event that they could present a problem later.

4.2 Parallel SMVGEAR Simulations

Table 3 shows total simulation time and scalability for 1, 4, 9, 18 and 25 PE cases. Also indicated is the degree of load imbalance within the chemistry between the slowest PE and the mean of all PEs.

Table 3 Scalability trend with the SMVGEAR solver.

# PEs	Time (s)	Scalability (%)	CHEM imbalance (%)
1	255K	≡ 100	---
4	70K	91	11
9	33K	86	16
18	18K	78	20
25	14K	73	20

The load imbalance even for the 25 PE case is only on the order of 20%, and the scalability of the overall code is good even for 25 PEs. The chemistry load imbalance accounts for much of the scalability loss (since chemistry accounts for such a large fraction of the calculation), except for the 25 PE case, in which other modules may have increased influence

4.3 Parallel EBI Simulations

Table 4 shows the performance of the various scientific modules for 1,9 and 18 PE runs using the EBI solver.

Clearly some modules scale well (chemistry (CHEM) and vertical advection (ZADV)), and others poorly. Among those that scale poorly horizontal diffusion (HDIFF) uses only a small fraction of the total time and so does not play a role in overall scalability degradation. With the information in the table above we can experiment with what-if scenarios to see the benefit of improving the scalability in a given scientific module. HADV is clearly the largest cause for the drop in scalability, followed by VDIF and then by non-SCIPROC computation. This last measure is apparent from the difference between the "Sum" entry, which is the sum of the 5 scientific processes called from SCIPROC, and the "Total" entry, which is the total time of the entire CMAQ simulation, and which includes all of the initialisation.

We shall now concentrate on isolating the cause of the HADV scalability drop. Further timing calls were inserted inside the HADV subroutine and its daughters. In particular, these were placed to return

1. the total times inside selected daughter subroutines,

2. the times spent for MPI communication calls and
3. the times spent for disk accesses.

Table 4 Scalability trend with the SMVGEAR solver differentiated by scientific module. “Total time” column is the average over all PEs of the total time that each PE has spent in a particular scientific module. The row “Total” contains the total time including that spent outside of the scientific processes.

Module name	Total time (s)	Fraction	Scalability (%)
1 PE			
CHEM	7370	0.34	≡ 100
HADV	7610	0.35	≡ 100
HDIFF	633	0.029	≡ 100
VDIF	5310	0.25	≡ 100
ZADV	733	0.034	≡ 100
Sum	21656	≡ 1	≡ 100
Total	24920	---	≡ 100
9 PE			
CHEM	821	0.24	99.7
HADV	1644	0.48	51
HDIFF	139	0.04	50
VDIF	727	0.21	81
ZADV	81	0.02	100
Sum	3412	≡ 1	70
Total	4255	---	65
18 PE			
CHEM	418	0.18	98
HADV	1272	0.56	33
HDIFF	87	0.04	40
VDIF	457	0.20	64
ZADV	40	0.02	100
Sum	2274	≡ 1	53
Total	2821	---	49

Table 5 shows some of these times.

For the 1 PE case we see that actual computation (HADV Work) accounts for most of the time, and that most of this is spent inside the HPPM subroutine, which is a 1-dimensional, row- or column-wise advection calculation using the piecewise parabolic method. For the 9 and 18 PE cases communication dominates the cost, being larger than actual computation times. “HADV Work” scales well with number of PEs. “HADV Comm”, the communication cost, scales very poorly. The source of most of the communication cost is within HPPM, as “HPPM Comm” accounts for most of “HADV Comm”. Disk file access also scales very poorly, and if and when the communication scalability problem is solved, it too will have to be dealt with.

Table 5 Time spent in HADV module, differentiated by function. “HADV Work” is time spent in actual computation. “HADV Comm” is time spent in MPI communication. “HADV Disk” is time spent on disk file access. “HPPM total” is total time spent in the subroutine HPPM, and “HPPM Comm” is that part of MPI communication that occurs within HPPM.

Name	Total time (s)
1 PE	
HADV Work	6780
HADV Comm	209
HADV Disk	585
HADV Sum	7574
HPPM total	4800
HPPM Comm	168
9 PE	
HADV Work	497
HADV Comm	903
HADV Disk	280
HADV Sum	1680
HPPM total	1189
HPPM Comm	797
18 PE	
HADV Work	264
HADV Comm	872
HADV Disk	258
HADV Sum	1394
HPPM total	978
HPPM Comm	767

HPPM is called numerous times within the HADV module, and on each call calculates the advective transport in the x or y direction by the x or y component of velocity. We expect its communication costs to depend on the orientation of the grid partitioning. In a 9 PE simulation with the grid partitioned 3 ways in each direction, the mean cost of HPPM communication per advection time step is about 0.7s whether HPPM is working in the row- or column-wise direction. However for a 9 PE simulation which uses a grid partitioned 9-ways in the row-wise direction and not at all in the column-wise direction, the cost of HPPM in the row-wise direction rises to 2.2 s but drops to 0.02 s in the column-wise direction, for which no actual inter-process communication is necessary.

5. CONCLUSIONS AND RECOMMENDATIONS

5.1 SMVGEAR Simulations

When used with the SMVGEAR solver CMAQ v4.5 scales fairly well even up to 25 PEs. Beyond this, we have found that load imbalance in the chemistry and possibly scalability degradation from HADV cause the overall scalability to drop below 70%. Load balancing for chemistry alone can be implemented fairly easily since chemistry is calculated on a per-grid-cell basis. A producer/consumer model in which equal workloads are parceled to the PEs can result in a very high scalability.

5.2 EBI Simulations

Scalability of 65% is seen with EBI even with 9 PEs. The dominant cause of scalability decrease is communication within the HPPM subroutine of the horizontal advection scientific module. Using information from Tables 4 and 5, we calculate that reducing this communication time to that of the disk file access time would result in an overall scalability of 76% with 9 PE and 62% with 18 PE instead of the current 65% and 49% respectively. Methods to implement this might be (1) to restructure the HADV code to process blocks of rows/columns at one time instead of single rows/columns at a time or (2) to initially setup alternate MPI communicators, grid sub-domains and stencils aligned along the row and column directions and use these for horizontal advection. The former method would be easier to implement. The latter may require recoding of the stencil library to permit multiple stencils and to allow CMAQ to dynamically switch to using a different stencil during run time.

Other causes of scalability decrease which we have identified, and would be the next bottlenecks to address, are the disk accesses during horizontal advection and vertical diffusion.

6. ACKNOWLEDGEMENTS

We are grateful to the California Energy Commission for making possible this study and analysis.