Work Order No. 582-15-52944-05
Contract # 582-15-50417
Project #FY15-35

Final Report
CAMx Speed Improvements

PREPARED UNDER A CONTRACT FROM THE
TEXAS COMMISSION ON ENVIRONMENTAL QUALITY

The preparation of this report was financed through a contract from the State of Texas through the Texas Commission on Environmental Quality. The content, findings, opinions and conclusions are the work of the author(s) and do not necessarily represent findings, opinions or conclusions of the TCEQ.

Prepared for:
Jim MacKay
Texas Commission on Environmental Quality
121 Park 35 Circle MC 164
Austin, TX 78753

Prepared by:
Christopher Emery, Gary Wilson,
DJ Rasmussen, Greg Yarwood
Ramboll Environ
773 San Marin Drive, Suite 2115
Novato, California, 94998

July 2015

06-35854E

CONTENTS

TABLES

FIGURES

EXECUTIVE SUMMARY

Recent additions and updates to the Comprehensive Air quality Model with extensions (CAMx) have increased demands on computer resources and have extended model runtimes. The Texas Commission on Environmental Quality (TCEQ) plans to conduct seasonal modeling with particulate matter (PM) and high resolution grids, all of which increase computational demands. The objectives of this project included (1) identifying areas of the CAMx code where improvements would likely have the most impact on model speed, (2) developing and testing various methods to achieve speed improvements, and (3) documenting the speed and accuracy impacts of these modifications using a TCEQ modeling dataset.

Ultimately we implemented several modifications that resulted in the following speed improvements on two of our computer systems using the TCEQ 2012 modeling database with halogen chemistry (CB6r2h) and the Plume-in-Grid (PiG) module:

- 15-30% speed improvement when compiled using the Portland Group (PGF) compiler;
- 45-50% speed improvement when compiled using the Intel (IFORT) compiler.

The project was conducted under two phases. In Phase 1, we identified areas of the model most needing speed enhancements according to analyses conducted using third-party code profiling tools. Two issues were immediately apparent from our analysis:

- TCEQ should use the “K-theory” vertical diffusion in lieu of the ACM2 option. This is a runtime option that requires no alternative input data or model compilation, and generates practically identical results in much less time.
- An unnecessary PGF compiler flag “-Mconcur=nonuma” was found to dramatically increase CAMx run times with PGF v13.4. TCEQ should remove this option in case their PGF compiler version is similarly affected by this flag.

Under Phase 1 we found that the chemistry solver, and specifically a single routine that calculates reaction rates, comprises up to 50% of simulation time and that the addition of halogen chemistry slowed the model down disproportionately to the number of additional species and reactions. We also identified certain issues related to our implementation of OMP parallelization in CAMx.

Phase 2 addressed the highest priorities identified in Phase 1 according to available project resources and schedule. We implemented and tested several modifications to CAMx code, OMP parallelization and compiler flags, and found five specific changes that were most effective in improving model speed. We found that halogen chemistry (CB6r2h) derives more speedup than the standard CB6r2 mechanism, and that the modifications tend to align model speed performance across compilers and hardware. Impacts to peak ozone concentrations from these modifications were found to be well below 1 ppb in cases without PiG, to generally just below 1 ppb (and in some cases just exceeding 1 ppb) with PiG. We see no impacts to OMP scalability with these speed improvements.

We have provided the updated version of CAMx to TCEQ at the close of this project. We recommend that TCEQ conduct tests over their longer modeling periods to benchmark speed improvements and evaluate OMP/MPI parallelization combinations. Initial tests performed on the agency's computer cluster system, but with much more parallelization, indicate speed improvements of about 15%, which are consistent with our results. Greater speed improvements by roughly 8% are realized without the use of PiG, which is insensitive to speed improvements implemented in this project.

Changes to CAMx code and compilation flags will be incorporated into the next public release of CAMx. Based on results from this project, we recommend follow-on work to further improve CAMx speed performance.

1.0 INTRODUCTION

The Texas Commission on Environmental Quality (TCEQ) uses the Comprehensive Air quality Model with extensions (CAMx) for ozone State Implementation Plan (SIP) modeling of Texas' ozone non-attainment areas. The TCEQ is developing new modeling periods that extend over the entire summer season of 2012. Modeling will include PM chemistry and TCEQ is considering the use of very high resolution (1 km) grids over key areas of Texas. Recent science advancements in CAMx, such as halogen chemistry, have added additional chemical species and reactions to the gas-phase mechanism. All of these updates require more computer resources and extend the length of model runtimes relative to previous applications. Reducing the turnaround time of each simulation would increase the number of analyses that can be performed for the next round of SIP modeling.

1.1 Objectives

The purpose of this Work Order is to improve the efficiency and speed of CAMx. The project was conducted in two phases. In Phase 1, we identified areas of the model most needing speed improvements according to analyses conducted using third-party code profiling tools, similarly to previous projects conducted for TCEQ (Wilson and Johnson, 2010; ENVIRON, 2012). From these results, an initial strategy plan was developed that outlined our findings and set a prioritized list of model modifications. In Phase 2, we implemented and tested modifications to CAMx code, OMP parallelization and compiler flags, addressing the highest priorities listed in the strategy plan according to available resources and schedule. We have identified other areas for improvement that could be accomplished in follow-on work. The modified code has been transferred to TCEQ for further validation and testing on the TCEQ computer system.

1.2 Report Organization

This section describes the purpose of this work. Section 2 presents findings from Phase 1 of the project and lists our initial strategy plan based on those results. Section 3 describes specific CAMx code updates, rationale, and testing results under Phase 2 of the project. Section 4 presents our conclusions and recommendations for future work.

2.0 INITIAL PROFILING AND STRATEGY DEVELOPMENT (PHASE 1)

Several CAMx screening tests were run using a compiler-based speed profiler to identify specific model subroutines that should be targeted for specific efficiency improvements or that would potentially benefit from improved parallelization. The testing dataset was selected with the TCEQ project manager and includes nested grids, Plume-in-Grid (PiG), halogen chemistry, and particulate matter (PM). A Phase 1 strategy plan (Emery et al., 2015) was developed that documented these tests and developed a prioritized list of CAMx code modifications that was expected to achieve the most effective set of modifications for the available schedule and resources. After reviewing the strategy plan, TCEQ authorized us to proceed with Phase 2.

2.1 Profiling Analyses

2.1.1 Modeling Database

We downloaded the 2012 CAMx modeling database from the TCEQ FTP site[1] on March 10, 2015. This dataset spans the historical period of May 16 through June 30, 2012 and includes three nested grids with 36, 12, and 4 km horizontal grid resolution (Figure 1), all with 28 vertical layers extending from the surface to an altitude of roughly 15 km. The modeling domain is established on the standard National RPO Lambert Conic Conformal projection. The input datasets for this application include meteorology, emissions, initial/boundary conditions, land cover, and other ancillary chemical data needed to run CAMx. We did not develop or modify any datasets.

2.1.2 CAMx Profiling Tests

All CAMx profiling tests were conducted on a single isolated multi-core Linux server to remove any influences from extraneous CPU loads and network traffic:

- Quad 16-core AMD Opteron 6380 chipset, 2.5 GHz
- Linux/CentOS 6.3
- Portland Group Fortran90 (PGF90) Workstation v13.4
- MPICH v3.0.4

This is the same workstation employed for TCEQ's Near-Real Time ozone modeling projects (Johnson et al., 2013, 2015).

CAMx v6.20 (ENVIRON, 2015) was compiled using PGF90 with profiling invoked for OMP parallelization. Our current PGF90 license does not support profiling for MPI parallelization. We installed a trial license for the Portland Group's Cluster Development Kit (CDK) for PGF90 v15.3, which is advertised to support profiling for MPI. However, Portland Group subsequently informed us that bugs precluded MPI profiling and indicated that this was scheduled to be fixed in the next CDK release (v15.4). Therefore, MPI profiling was not performed.

[1] ftp://amdaftp.tceq.texas.gov/pub/TX/camx/basecase/bc12_12jun.reg3a.2012_wrf361_p2a_i2_a/

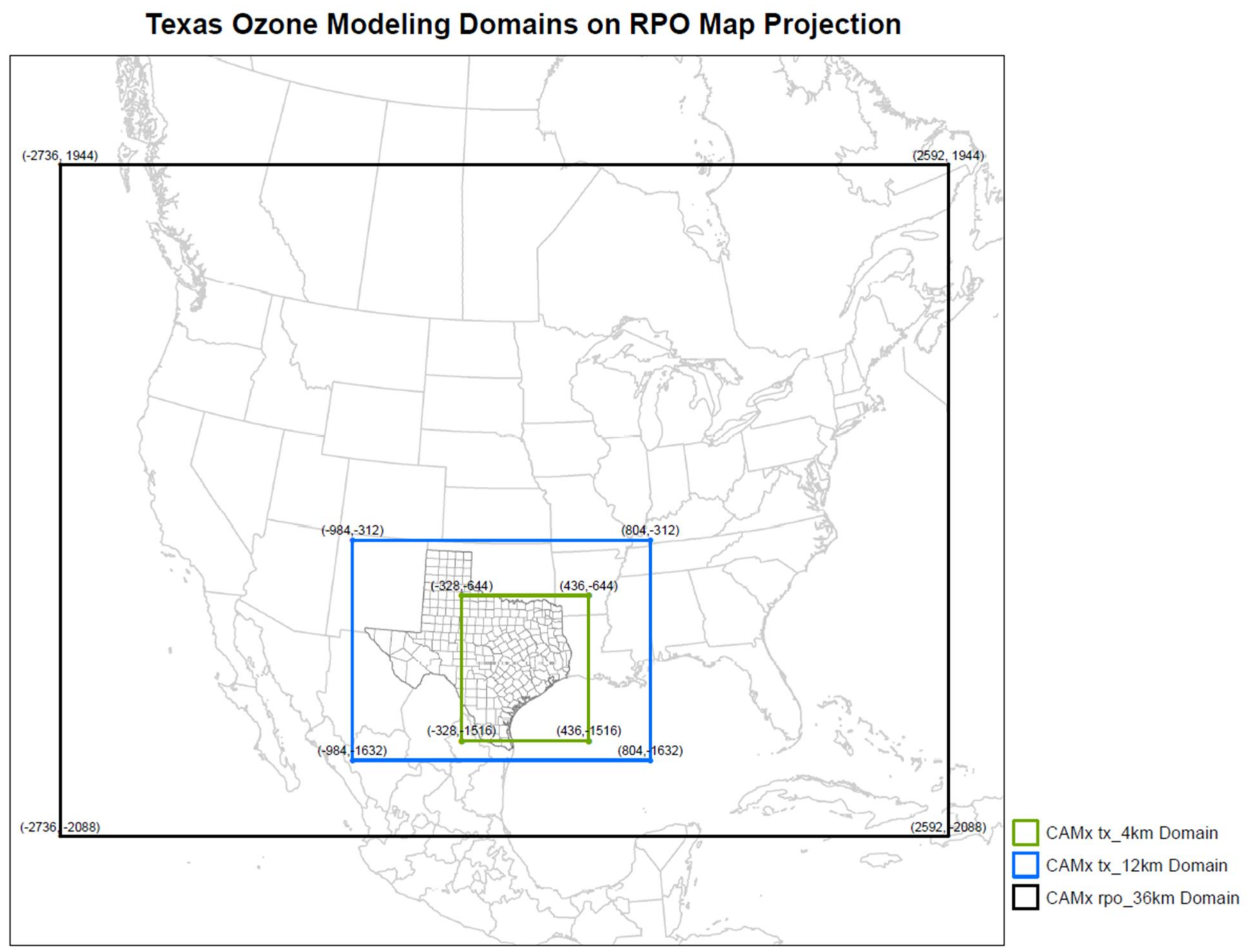

Figure 1. TCEQ modeling domains for the 2012 modeling database.[2]

The following three CAMx configurations were profiled under Phase 1:

1. May 17 (restart): 3 grids, PiG, ACM2 vertical mixing, chemistry CB6r2/CF (CB6r2 + PM), no compiler optimization, OMP parallelization (8 threads).
2. May 17 (restart): 3 grids, PiG, K-theory vertical mixing, chemistry CB6r2/CF (CB6r2 + PM), "O2" compiler optimization, OMP parallelization (8 threads)
3. May 17 (restart): 3 grids, PiG, k-theory vertical mixing, chemistry CB6r2h (no PM), "O2" compiler optimization, OMP parallelization (8 threads)

Run times during the first simulation day (May 16) may not be representative of other days of the episode due to chemical spin-up from initial conditions. Therefore, we conducted code profiling for May 17 to separate reported processor times from the chemical spin-up.

[2] https://www.tceq.texas.gov/airquality/airmod/data/domain

2.1.3 Results

2.1.3.1 Run 1

Table 2-1 presents Run 1 timing results from OMP thread 0 (consisting of all model processes). The sum of all processes and individual routines listed comprised 95% of the 13,702 second total CPU time for thread 0.

Chemistry used the majority of time, with more than 23% spent in gas-phase chemistry versus less than 3% in PM chemistry. The EBIRATE routine alone used one-third of time spent in gas chemistry. EBIRATE calculates linear combinations of chemical rates for all reactions in the gas-phase mechanism and is implemented in a manner that accesses the chemical rate array non-sequentially. The EBIRXN routine provides an interesting comparison because EBIRXN performs similar operations to EBIRATE but EBIRXN accesses memory more sequentially and consumes less CPU time (4% as compared to 8%). Non-sequential memory access may be a cause of inefficiency in EBIRATE. We assigned a high priority to making EBIRATE more efficient.

Diffusion consumed the second most CPU time after chemistry. The majority of time (11%) was spent in the ACM2 vertical diffusion solver whereas horizontal diffusion used less than 5%. Using the default K-theory vertical diffusion should reduce total time in diffusion to less than 10% of total run time.

The process labelled “OMP” includes Fortran library functions to manage parallelization. OMP added 13% of overhead to the run time for thread 0. About 5% of time was spent in “barrier” functions, indicating load imbalances where completed threads wait for the slowest threads to finish. We identified 3 routines with OMP load imbalances: PIGDRIVE, TRIDIAG, and ZADVEC. By far the largest OMP load imbalance existed in PIGDRIVE which applies OMP parallelization to the loop over puff chemistry in first half of the routine but not to the loop over puff growth and dumping in the second half. The load imbalance in the puff chemistry loop was mostly likely caused by skipping puffs because they are inactive, outside the current grid of interest, or chemistry is bypassed or highly simplified for strategic reasons. Prior to the effort to accelerate PiG chemistry (Emery et al., 2013), implementing OMP just for chemistry was an effective strategy for PIGDRIVE. Now that PiG chemistry runs quickly the parallelization of the growth/dumping loop should be considered. TRIDIAG and ZADVEC are called in vertical transport (advection and diffusion) where load imbalances most likely result from skipping grid columns where the solution is replaced by results from an underlying nested grid, and secondarily from variations in solution times among different grid columns.

The process labelled “System” includes Fortran intrinsic math functions (e.g., logarithms) and management processes such as dynamic allocation. System routines took 10% of total runtime. It was not originally clear how this process could be improved for speed, but in Phase 2 we identified a compiler option (IEEE) that significantly increases runtime. This option forces math expressions to be consistently calculated among different compilers, resulting in identical CAMx results between PGF90 and Intel compilations at the cost of execution speed.

Table 2-1. Run 1 process and routine CPU time for OMP thread 0. Routines parallelized with OMP are noted.

Process	Routine	OMP	Time (s)	Time (% of total)
Chemistry 26%	ebirate2	●	1,033	8%
	ebisolv	●	495	4%
	chemdriv	●	415	3%
	hr_hox2	●	385	3%
	ebirxn2	●	338	2%
	hr_nox2	●	214	2%
	hr_nxy2	●	106	<1%
	hr_pan2	●	93	<1%
	calcact	●	90	<1%
	ktherm	●	79	<1%
Diffusion 16%	matrix	●	803	6%
	diffus	●	624	5%
	vdiffacm2	●	528	4%
	tri	●	132	1%
OMP 13%	_mp_taskv2_init_contexts		307	2%
	_mp_barrierw		283	2%
	_mp_barrier		260	2%
	_mp_get_schedule		163	1%
	_mp_p2		162	1%
	_mp_barrierr		159	1%
	_mp_cslave		157	1%
	_mp_cdecl		94	<1%
	_mp_create_team		85	<1%
	_mp_get_bind		76	<1%
V Advection 11%	vrtslv	●	620	5%
	tridiag	●	402	3%
	zadvec	●	252	2%
	zrates	partial	72	<1%
System 10%	__mth_i_dpowd		208	2%
	_int_free		188	1%
	__mth_i_dexp		187	1%
	__mth_i_dexp2		172	1%
	__c_mzero4		164	1%
	__mth_i_dlog2		156	1%
	__mth_i_exp		122	<1%
	_int_malloc		74	<1%
	__c_mcopy4		72	<1%
H Advection 9%	hadvppm	●	933	7%
	xyadvec	●	237	2%
PiG 5%	pigdrive	partial	490	4%
	virtdump		103	<1%
TUV 2%	drvtuv		138	1%
	pp2str		83	<1%
Output 1%	aggreg	●	72	<1%
	average		111	<1%
Emissions 1%	Emiss	partial	137	1%

2.1.3.2 Run 2

Table 2-2 presents Run 2 timing results from OMP thread 0 (consisting of all model processes). The sum of all processes and individual routines listed comprised 93% of the 12,763 second total CPU time for thread 0. Total run time decreased 7% relative to Run 1.

Chemistry continued to use the most time, increasing to more than 28% spent in gas-phase chemistry versus less than 3% in PM chemistry. EBIRATE remained one-third of time spent in gas-phase chemistry and the absolute time spent in EBIRATE increased over Run 1 indicating that compiler optimization was ineffective, if not detrimental for that routine.

OMP and System processes both increased relative to Run 1. OMP added 14% of overhead to the run time for thread 0, with 5% of time spent in “barrier” functions. System routines took 11% of total run time. With the removal of ACM2, run time spent in diffusion dropped to 8% (as anticipated) and diffusion ranked below horizontal and vertical advection. The PiG process improved relative to Run 1 (possibly due to optimization) while the output process was relatively slower.

2.1.3.3 Run 3

Table 2-3 presents Run 3 timing results from OMP thread 0 (consisting of all model processes). The sum of all processes and individual routines listed comprised 92% of the 15,519 second total CPU time for thread 0. Total run time increased 22% relative to Run 2.

Chemistry continued to take the most time with gas-phase chemistry increasing to 48% of total run time with the introduction of halogen chemistry. EBIRATE took nearly half of the time spent in gas chemistry (a quarter of total run time) due to the larger number of reactions and species in the halogen version. EBIRATE consumed four times more CPU than EBIRXN (21% vs. 5%) suggesting that revisions to EBIRATE could realize substantial speedup for larger chemical mechanisms such as CB6r2h. OMP and System processes both decreased relative to Run 2. Run times for all other processes remained roughly similar to Run 2.

Subsequently we discovered that PGF v13.4 slowed the model considerably relative to our legacy compiler (v8.0). We tracked the cause to a particular PGF compiler flag (“-Mconcur= nonuma”). This flag was left over from earlier versions of CAMx when the compilation was set to build static executables, whereas now the compilation builds dynamic executables. Removal of this flag has no impact on model results other than speed. It is unclear at which PGF version (between v8 and v13) this flag began to negatively impact model speed for dynamic builds. We have removed this flag from all subsequent testing documented in this report, and have included this change in the updated model delivered to TCEQ.

Profiling for Run 3 was repeated with the “nonuma” flag removed. Table 2-4 presents timing results from OMP thread 0 (consisting of all model processes). The sum of all processes and individual routines listed comprised 93% of the 11,952 second total CPU time for thread 0. Total run time decreased 23% relative to Run 3 with “nonuma”. Although the order and relative time spent in each process remained similar to Table 2-3, the reduction in time spent in

Table 2-2. Run 2 process and routine CPU time for OMP thread 0. Routines parallelized with OMP are noted.

Process	Routine	OMP	Time (s)	Time (% of total)
Chemistry 31%	ebirate2	●	1252	10%
	Ebisolv	●	540	4%
	ebirxn2	●	492	4%
	hr_hox2	●	462	4%
	Chemdriv	●	377	3%
	hr_nox2	●	228	2%
	hr_nxy2	●	104	<1%
	hr_pan2	●	92	<1%
	Calcact	●	72	<1%
	Khetero	●	64	<1%
OMP 14%	_mp_barrier		333	3%
	_mp_taskv2_init_contexts		307	2%
	_mp_cpenter		215	2%
	_mp_p2		166	1%
	_mp_has_running_subtasks		159	1%
	_mp_barrierw		141	1%
	_mp_barrierr		134	1%
	_mp_cdeclp		94	<1%
	_mp_get_blist		84	<1%
	_mp_create_team		74	<1%
System 11%	__mth_i_dpowd		208	2%
	_int_free		205	2%
	__mth_i_dexp		168	1%
	__mth_i_dexp2		163	1%
	__mth_i_dlog2		153	1%
	_wordcopy_fwd_aligned		151	1%
	__mth_i_exp		133	1%
	_int_malloc		84	<1%
V Advection 11%	Zadvec	●	477	4%
	Tridiag	●	379	3%
	Vrtslv	●	348	3%
	Zrates	partial	71	<1%
H Advection 9%	Hadvppm	●	906	7%
	Xyadvec	●	224	2%
Diffusion 8%	Diffuse	●	540	4%
	Tridiag	●	379	3%
	Vdiffimp	●	116	<1%
PiG 3%	Pigdrive	partial	423	3%
Output 3%	Massum		169	1%
	Average		139	1%
	Aggreg	●	72	<1%
TUV 2%	Drvtuv		125	1%
	pp2str		89	<1%
Emissions 1%	Emiss	partial	130	1%

Table 2-3. Run 3 process and routine CPU time for OMP thread 0. Routines parallelized with OMP are noted.

Process	Routine	OMP	Time (s)	Time (% of total)
Chemistry 48%	ebirate3	●	3250	21%
	hr_hox3	●	1028	7%
	ebisolv	●	869	6%
	ebirxn3	●	770	5%
	hr_nox3	●	606	4%
	chemdriv	●	367	2%
	hr_nxy3	●	259	2%
	hr_pan3	●	227	1%
	kphoto	●	62	<1%
OMP 12%	_mp_taskv2_init_contexts		330	2%
	_mp_barrier		309	2%
	_mp_cpenter		216	1%
	_mp_barrier2		167	1%
	_mp_has_running_subtasks		167	1%
	_mp_p2		166	1%
	_mp_threads_at_level		164	1%
	_mp_barrierw		161	1%
	_mp_create_team		136	<1%
	_mp_barrierr		105	<1%
V Advection 8%	zadvec	●	457	3%
	tridiag	●	364	3%
	vrtslv	●	343	2%
	zrates	partial	71	<1%
Diffusion 7%	diffuse	●	512	3%
	tridiag	●	364	3%
	vdiffimp	●	114	<1%
H Advection 7%	hadvppm	●	900	6%
	xyadvec	●	232	1%
System 5%	_int_free		184	1%
	__mth_i_dexp2		136	<1%
	__mth_i_dlog2		128	<1%
	__mth_i_dpowd		96	<1%
	_int_malloc		92	<1%
PiG 3%	pigdrive	partial	431	3%
Output 2%	massum		176	1%
	average		87	<1%
	aggreg	●	74	<1%
Emissions 1%	Emiss	partial	130	1%

Table 2-4. Process and routine CPU time for OMP thread 0 resulting from a rerun of Run 3 (Table 2-3) with the PGF "-Mconcur=nonuma" flag removed. Routines parallelized with OMP are noted.

Process	Routine	OMP	Time (s)	Time (% of total)
Chemistry 51%	ebirate3	•	2,715	23%
	hr_hox3	•	868	7%
	ebirxn3	•	659	6%
	hr_nox3	•	513	4%
	ebisolv	•	471	4%
	chemdriv	•	348	3%
	hr_nxy3	•	228	2%
	hr_pan3	•	196	2%
V Advection 9%	tridiag	•	316	3%
	zadvec	•	306	3%
	vrtslv	•	257	2%
	zrates	partial	92	<1%
Diffusion 9%	diffuse	•	539	5%
	tridiag	•	364	3%
	vdiffimp	•	114	<1%
H Advection 7%	hadvppm	•	651	5%
	xyadvec	•	188	2%
PiG 5%	pigdrive	partial	593	5%
OMP 5%	_mp_barrier		591	5%
System 4%	__c_mzero4		151	1%
	__mth_i_dexp2		136	1%
	__mth_i_dlog2		131	1%
	__mth_i_dpowd		91	<1%
Output 3%	massum		172	1%
	average		85	<1%
	aggreg	•	73	<1%
Emissions 1%	Emiss	partial	178	1%

OMP routines was the largest impact, from 12% to just 5% (about 40% of the total reduction in run time).

2.2 Prioritized List of Speed Improvements

At the close of Phase 1, we developed a list of recommended changes based on the profiling analyses described above and in accordance with our understanding of potential speed impediments at the time. We ranked the actions from highest to lowest priority considering ease of implementation and likely impact to run time. Actions toward the bottom of the list require greater effort and more fundamental changes to the CAMx code structure, but carry less certainty for speed improvements. Whereas our understanding of factors impacting CAMx speed have evolved with our analyses and tests conducted under Phase 2, recommendations beyond the first four remain relevant and should be considered for future improvements. Additional details for each item below are provided in the Phase 1 report (Emery et al., 2015).

1. We recommend that K-theory be used in lieu of ACM2. This is a runtime option that requires no alternative input data or model compilation. ACM2 adds significantly to CAMx run time but results in similar concentration patterns relative to the default K-theory option. Furthermore, ACM2 is not compatible with all CAMx Probing Tools, requiring the use of K-theory for any decoupled direct method (DDM) sensitivity or process analysis applications. The ACM2 solver was developed by EPA; we understand that EPA is improving the ACM2 solver and will distribute it with the release of CMAQ v5.1 in September 2015.
2. Revise EBIRATE to employ a more efficient strategy. Test several approaches to maximize efficiency. Improving the efficiency of EBIRATE is a high priority.
3. Add OMP parallelization to PIGDRIVE around the loop that performs puff growth and mass dumping to the grid.
4. Test alternative ways to parallelize loops in transport processes (vertical/horizontal advection and diffusion) to improve OMP load balance and efficiency, if possible.
5. Apply F90 vector methods to variable assignments where possible, rather than using explicit loops and array index pointers.
6. Improve MPI load balancing by considering spatial distributions of PiG sources and potentially other factors. Run times for specific MPI sub-domains may be influenced by the workload required to process PiG puffs.
7. Restructure the order of dimensions in major multi-dimensional variable arrays (concentration, meteorology, and other fields) for improved memory caching in local routines.
8. Revise process splitting order (in combination with restructuring the dimensions of variable arrays). The goal would be to minimize the amount of memory caching and time taken in assigning local variable arrays from the major concentration and meteorological arrays. A more efficient approach would be to group all processes that operate on the same spatial dimension.

The modification noted in point (8) above would likely translate to improved MPI scaling efficiency as well by reducing the number of overlapping (shared) grid cells among MPI sub-domains. Overlapping cells are used as internal "boundary conditions" through which chemical mass is passed among the sub-domains. The number of overlapping cells is currently 5, and this is based specifically on the process order currently implemented in CAMx; 5 overlapping cells are needed to ensure that an MPI application results in exactly the same concentration fields as a serial run. Reordering the physical processes in the model could conceivably reduce the overlap to 2 or 3.

As an example, consider a moderate grid of 120x90 grid cells with MPI applied over 16 cores, or 4x4 MPI sub-domains. The addition of 5 overlap cells surrounding each sub-domain results in 67% more grid cells to solve (i.e., 67% additional overhead). This translates to a best scaling efficiency of 60% for this configuration, meaning a run on 16 cores is at best equivalent to 9.6 cores (ignoring other forms of overhead involving inter-core network memory passes and MPI management, which in reality further reduce scalability). Looking at this another way, if that run takes 16 hours/day on 1 core, 100% scaling efficiency would theoretically take 1 hour on 16 cores, but with 5 overlap cells would take at best 1.7 hours. Doubling MPI cores to 32 further reduces scaling efficiency to 49% (effectively 15.7 cores); the 16 hour/day serial run would theoretically take 0.5 hours, but with 5 overlap cells would take 1 hour. If overlap cells could be reduced to just 2 in each dimension, scaling efficiency would be 80% for 16 cores (12.8 effective cores) and 73% for 32 cores (23.4 effective cores). In this case, a 16 hour serial run would be reduced to 1.25 hours with 16 cores and 0.7 hours with 32 cores.

3.0 CODE IMPROVEMENTS AND TESTING (PHASE 2)

We carefully considered, implemented and tested the effects of several targeted modifications to CAMx v6.20 based on the Phase I Strategy Plan. Impacts to model speed were compared to the original code for each modification. Our approach focused on addressing the highest priority issues to the extent that the project schedule allowed.

We conducted testing using two datasets: (1) certain interim assessments of code and compiler option changes were performed using the standard CAMx test case that is distributed with the model at www.camx.com; (2) other assessments and all final testing were performed using the TCEQ modeling database employed in Phase 1. Testing included variable OMP and MPI parallel processing to gauge efficiency gains and to verify consistent model output, with appropriate quality assurance steps and code review. Our goals for this phase included improving model speed for a set number of processors, and to the extent possible, extend speed gains out to larger numbers of processors than can be currently realized (i.e., improve parallelization “scalability”).

3.1 Code and Compilation Modifications

3.1.1 Chemical Solver Efficiency and PiG Load Imbalances

Our early efforts under Phase 2 focused on improvements to the EBIRATE chemistry routine and OMP imbalances related to sparse PiG puff operations (chemistry, growth, dumping).

EBIRATE updates net production and loss rates for all gas species at each chemical time step by calculating linear combinations of individual reaction rates in a manner that accesses the chemical rate array non-sequentially. At the time we expected non-sequential memory access to be a cause of inefficiency. EBIRATE is one of several chemistry subroutines that are computer-generated by the Chemical Mechanism Compiler (CMC). The CMC codes EBIRATE in a straightforward manner without any specific strategy for accessing memory efficiently.

We considered several approaches to improve EBIRATE. First, we replaced EBIRATE with an optimized linear algebra routine available in the Netlib LAPACK library (http://www.netlib.org/lapack/), but this resulted in slower execution speed. Second, we wrote a new vector multiplication routine specific to EBIRATE, but that was also slower. Third, we restructured array indices and loop order, but that had only a marginal impact on EBIRATE speed. Therefore, these approaches were dropped from further consideration.

Overhead associated with OMP parallelization is exacerbated by load imbalance among processor threads. This can occur when successive iterations through a parallelized loop have very different workloads, such as the extreme example when some iterations have nothing to do (e.g., an empty puff or a grid cell that is skipped because it contains a nest) but other iterations must perform a heavy workload (e.g., chemistry). We expected that PiG processes in particular are a source of significant OMP load imbalance because PiG memory arrays become quite sparse as puff populations grow and individual puffs are deactivated at different times. We developed and implemented a new routine to condense the PiG variable arrays at each time step, thereby maintaining active puffs in sequentially-ordered memory structures. However, only marginal speed impacts were realized in tests conducted with the 2012 TCEQ dataset. This modification was dropped from further consideration in this project, but we may continue to test it for a variety of PiG applications in the future.

Another potential issued identified in Phase 1 was the lack of OMP parallelization applied to the PiG growth and dumping section of PIGDRIVE. Implementing OMP around the puff growth/diffusion loop required substantial modifications to the PIGDRIVE routine to allow common memory structures (i.e., three-dimensional gridded concentrations) to be mathematically combined with OMP thread-private puff variables (i.e., mass to be dumped). Again, no significant speedup resulted from this change and so it was dropped from further consideration in this project. However, it may be implemented in future efforts to expand OMP to other areas of the code.

3.1.2 Additional Modifications

During the course of our investigations under Phase 2, several additional code modifications were implemented and tested. Each is described below.

3.1.2.1 IEEE and Mixed-Mode Math

We found that CAMx speed is impacted by a particular compiler flag that forces math expressions to be calculated using standard IEEE methods (i.e., –Kieee in PGF90 and -mieee-fp in IFORT). Without these flags, compilers are free to calculate math expressions using their own specific and often optimized strategies. Moreover, each compiler handles “mixed mode” math differently (e.g., real = real/integer, real = real*dreal, real = real**integer)[3]. Removal of IEEE flags can speed up the model, but leads to differences in mathematical calculations (particularly for mixed mode math) and thus model results. Up through v6.20, the CAMx “makefile” has included IEEE flags by default to ensure that nearly identical results are achieved across different compilers.

Elimination of mixed mode math throughout CAMx would reduce the dependency on IEEE flags (e.g. converting real=real/integer to real=real/float(integer)). Most instances are properly handled in CAMx, but it would take an extensive and concerted effort to review the entire model code to catch improper mixed mode statements. Several changes involving mixed mode algebra were made in the EXPTBL routine (which computes rate constants for gas-phase chemistry) and this did reduce dependency on the IEEE compiler flags. Tests suggest that the most important issues are with mixed single/double precision math. For one such case in a dry deposition routine IFORT resulted in a NaN when the IEEE flag was removed, but PGF completed successfully. The CAMx code was reviewed for such improper mixed single/double precision statements and fixed where necessary.

We plan to complete a systematic review of the entire CAMx code for mixed mode math in the future, although test results described below suggest that remaining differences from removing

[3] “dreal” refers to a double-precision real variable, “**” refers to raising to a power.

IEEE flags are small for ozone when PiG is not used. Larger but short-lived differences occur with PiG due to small changes in puff behavior (e.g., when and where mass dumping to the grid occurs). Larger differences may also occur for some PM species, particularly nitrate, and this is a result of how the ISORROPIA thermodynamic portioning algorithm is designed. The IEEE option is now an option in the CAMx makefile, and by default is not engaged.

3.1.2.2 EBI Solver Convergence

One reason why halogen chemistry runs more slowly than CB6r2 is that the EBI solver converges more slowly in high model layers. The solver was converging halogen species to 0.1% even when they had very small concentrations. We revised the EBI solver convergence criterion to be more tolerant of relative error when absolute error is smaller than 1×10^{-8} ppm. In tests described below, run time for the halogen mechanism was about 20% faster with this change, but nearly unchanged for CB6r2. Additional speedup for halogen chemistry may require EBI solver customization.

3.1.2.3 Streamlining EBIRATE

In CAMx v6.20 EBIRATE calculates terms for several species (NO, NO2, O3, OH, HO2, O1D, O, NO3, N2O5, PAN, C2O3, HONO, PNA) that the EBI solver does not need because these species are solved more efficiently by other EBI solver subroutines. These terms are only needed to calculate rate constant sensitivity when the Decoupled Direct Method (DDM) probing tool is invoked and is instructed to track these specific sensitivities. Commenting out these terms in CB6r2 demonstrated some speedup and no loss of accuracy (as expected). A more sophisticated and permanent solution is needed to modify the CMC and produce a dedicated subroutine that calculates these terms only when they are needed for DDM rate constant sensitivity.

3.1.2.4 Revised OMP Loop Scheduling and Collapsing

In CAMx v6.20 all OMP parallel do loops are scheduled as “dynamic”. This means that threads are allocated to cores dynamically as the loop executes, resulting in good load balance at the expense of overhead for managing dynamic scheduling. The “static” option incurs much lower overhead than “dynamic”. Loops with inherently good load balance (i.e., all loop iterations have the same computational load) should be scheduled as “static”, meaning loop iterations are distributed evenly to cores when loop execution starts. A middle ground is called “guided”, but tests with that option resulted in the slowest run time.

Load imbalances within CAMx OMP loops occur mostly when operations are skipped for grid cells covering an underlying nested grid (this happens for chemistry, vertical advection and diffusion). A more extreme issue exists for OMP looping in wet scavenging where non-raining grid columns are skipped, and for OMP looping over sparse PiG vectors as described in Section 3.1.1. These loops must remain scheduled as “dynamic”. In the future, we may consider eliminating the nested grid checks to skip computations, and scheduling all such loops as “static”, thus trading more work for better load balance and potentially improving OMP scalability to larger numbers of threads.

In CAMx v6.20 only the outermost loops are scheduled for OMP parallelization, offering fewer possible threads to distribute over cores. The OMP “collapse(2)” option can be applied to two immediately nested loops (outer and inner) so that they are scheduled together and act as a one loop over a single vector.

Based on test results for individual loops, we updated the OMP scheduling as follows (changes indicated in bold):

```
aggreg.f:   c$omp do schedule(static)
chemdriv.f: c$omp do schedule(dynamic) collapse(2)
diffus.f:   c$omp do schedule(dynamic)
drydep.f:   c$omp do schedule(static)
emiss.f:    c$omp do schedule(static)
wetdep.f:   c$omp do schedule(dynamic)
xyadvec.f:  c$omp do schedule(static)
zadvec.f:   c$omp do schedule(dynamic)
zrates.f:   c$omp do schedule(static)
```

Additionally, we added OMP loop scheduling in DRVTUV, which calculates in-line cloud and aerosol adjustments to clear-sky photolysis rates for each grid cell. This is another case for potential load imbalances as iterations are skipped for night conditions and the workload is reduced for non-cloudy grid cells.

```
drvtuv.f:   c$omp do schedule(static) collapse(2)
```

3.1.3 Interim Speed and Scalability Testing

The four modifications described in Section 3.1.2 were tested to evaluate impacts to model speed and OMP scalability. Interim tests were conducted for the second day of the 2-day CAMx test problem distributed with the model at www.camx.com. The domain consists of a relatively small (68×68) master grid at 36 km resolution covering the US Midwest, and a nested grid (92×115) at 12 km resolution covering the upper Midwest. Both grids have 16 vertical layers. PiG was turned off for the speed tests but turned on for OMP scalability tests. Two gas-phase chemical mechanisms were tested: CB6r2 and CB6r2h, both without PM.

Speed and accuracy tests were conducted on a dual 12-core Intel Xeon X5660 chipset (2.8 GHz) using 6 OMP threads for each run, but no MPI parallelization. Runs were conducted with PGF and IFORT compilations. Three runs were performed simultaneously with local I/O to minimize impacts from network latency. Model execution speed was metered using the Linux “/usr/bin/time” utility.

Tables 3-1 and 3-2 show the run time and accuracy results for (1) groups of CAMx updates and (2) removal of the IEEE compile flag. The run labelled “CAMx v6.20” is the unmodified version of the model. The modifications labelled “updates” include changes to EBI solver convergence and OMP loop scheduling. The modification labelled “trim” refers to EBIRATE streamlining to remove DDM rate sensitivities.

Table 3-1. CAMx speed test results using CB6r2 with PGF and IFORT compilers.

CB6r2 (no PM)	PGF		IFORT	
	Seconds	Percent	Seconds	Percent
CAMx v6.20, IEEE on	1329	100%	1521	100%
CAMx updates, IEEE on	1262	95%	1322	87%
CAMx updates, IEEE off	1125	85%	882	58%
Updates & trim, IEEE off	1028	77%	826	54%

Table 3-2. CAMx speed test results using CB6r2h with PGF and IFORT compilers.

CB6r2h (no PM)	PGF		IFORT	
	Seconds	Percent	Seconds	Percent
CAMx v6.20, IEEE on	2947	100%	3385	100%
CAMx updates, IEEE on	2273	77%	2308	68%
CAMx updates, IEEE off	2036	69%	1678	50%

Our findings from these results are:

1. Both the "updates" and removal of IEEE compiler flags improve speed;
2. Halogen CB6r2h derives more speedup from the updates than CB6r2;
3. IFORT derives more speedup from removing IEEE flags than PGF;
4. Removing IEEE flags appears to be safe after making the updates;
5. The "trim" modification gives some speedup but requires a CMC update to be automated;
6. IFORT ends up with a substantial (~20%) speed advantage over PGF.

Table 3-3 lists the largest differences in ozone (ppm) for the CB6r2 cases (Table 3-1). The reference case is chosen as "CAMx updates, IEEE on" with PGF because one of the CAMx updates changed EBI solver accuracy. Maximum differences of up to ~1×10^{-4} ppm (0.1 ppb) are considered acceptable. We also reviewed results for species other than ozone and found nothing alarming.

Table 3-4 shows the average number of iterations taken by the EBI chemistry solver, over all cells of both grids, for CB6r2 and CB6r2h in the tests described above. The EBI solver convergence update had only a small impact for CB6r2, but the update greatly reduced the number of iterations for CB6r2h leading to model speedup. However, CB6r2h continued to require more iterations than CB6r2 after the update.

Table 3-3. CAMx ozone accuracy test results using CB6r2 with PGF and IFORT compilers. The accuracy metric is the maximum difference in ozone (ppm) from the reference case.

CB6r2 (no PM)	PGF		IFORT	
	Grid 1	Grid 2	Grid 1	Grid 2
CAMx v6.20, IEEE on	9.77×10^{-5}	1.05×10^{-4}	9.76×10^{-5}	1.05×10^{-4}
CAMx updates, IEEE on	reference	reference	3.50×10^{-5}	9.58×10^{-5}
CAMx updates, IEEE off	2.44×10^{-6}	4.28×10^{-6}	3.61×10^{-5}	6.64×10^{-5}
Updates & trim, IEEE off	2.46×10^{-6}	4.28×10^{-6}	3.61×10^{-5}	6.64×10^{-5}

Table 3-4. Average number of iterations taken by the CAMx EBI chemistry solver by grid, for CB6r2 and CB6r2h.

Mechanism	Grid 1	Grid2
CB6r2 – original EBI solver	6.5	5.0
CB6r2 – updated EBI solver	5.7	4.7
CB6r2h – original EBI solver	16.6	10.4
CB6r2h – updated EBI solver	7.7	5.7

OMP scalability tests were conducted on a dual 12-core Intel Xeon X5660 chipset (2.8 GHz) using PGF and IFORT builds for both the unmodified CAMx v6.20 (labelled “STD”) and all code updates described in Section 3.1.2 (labelled “UPD”). MPI parallelization was excluded. Each run was performed sequentially with no extraneous workload that might confound the results. Figure 3-1 and Table 3-5 show the results of OMP scalability tests using CB6r2 without PiG invoked. The term “effective threads” indicates the net equivalent speedup for the given number of actual assigned threads due to un-parallelized areas of the model and various impediments from overhead processes. A perfectly scaling code would align along the 1:1 line in Figure 3-1. Although results were similar among CAMx and compiler versions, the IFORT STD case scaled best perhaps because it is the slowest (least efficient floating point execution).

OMP scalability tests were repeated on the same machine for CB6r2 with PiG invoked. Results are shown in Figure 3-2 and Table 3-6. A notable drop in scalability occurred with the introduction of PiG. However, the IFORT update case resulted in odd performance for unexplained reasons.

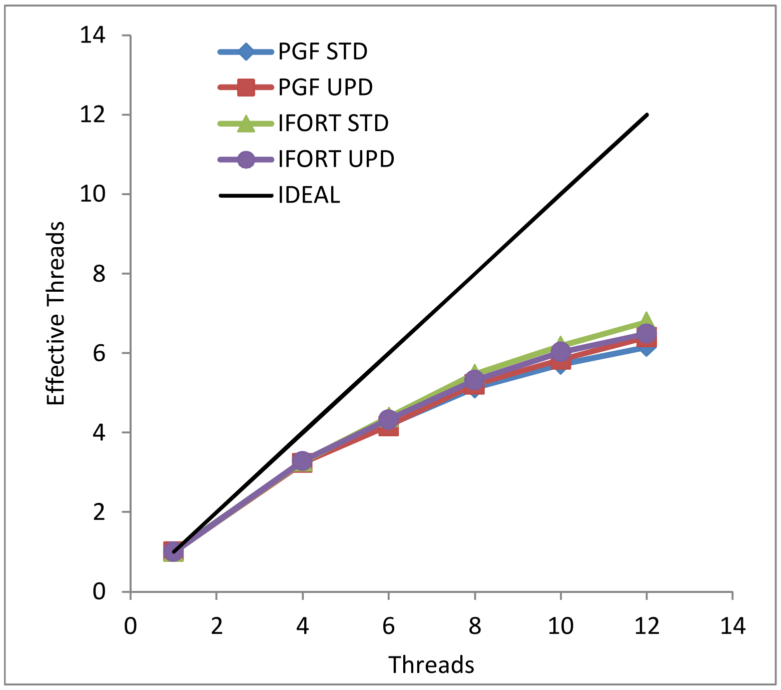

Figure 3-1. OMP scalability test results for the unmodified CAMx v6.20 (STD) and the modified code (UPD) for both PGF and IFORT compilers. Tests used CB6r2 (without PM) and without PiG invoked. Tests did not include MPI parallelization. The "IDEAL" case represents perfect parallel scalability.

Table 3-5. OMP scalability test results shown in Figure 3-1.

	PGF				IFORT			
	CAMx v6.20		CAMx Updates		CAMx v6.20		CAMx Updates	
Threads	Sec	Eff Threads	Sec	Eff Threads	Sec	Eff Threads	Sec	Eff Threads
1	4429	1.0	3786	1.0	5797	1.0	3065	1.0
4	1354	3.3	1170	3.2	1770	3.3	933	3.3
6	1058	4.2	908	4.2	1323	4.4	708	4.3
8	865	5.1	726	5.2	1059	5.5	576	5.3
10	775	5.7	649	5.8	938	6.2	509	6.0
12	720	6.2	591	6.4	854	6.8	472	6.5

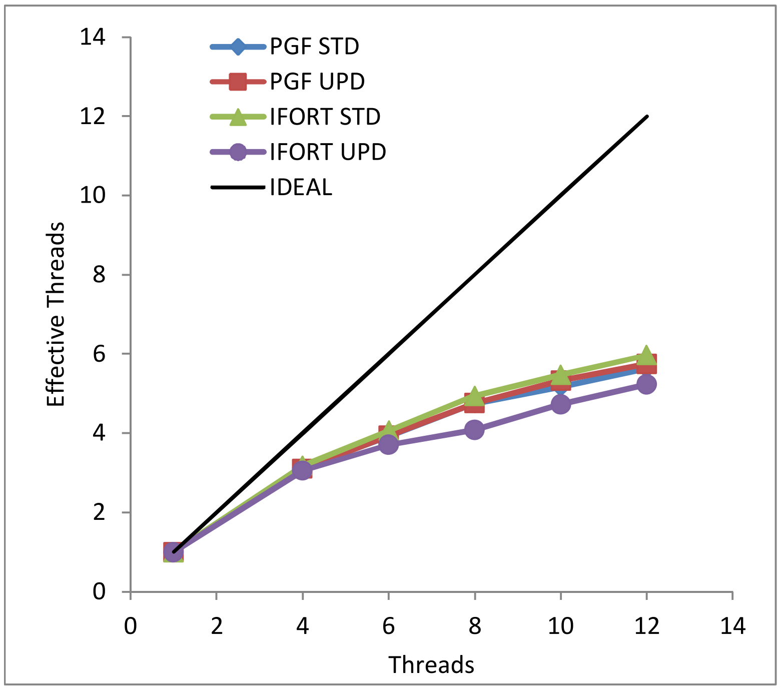

Figure 3-2. As in Figure 3-1, but with PiG invoked.

Table 3-6. OMP scalability test results shown in Figure 3-2.

	PGF				IFORT			
	CAMx v6.20		CAMx Updates		CAMx v6.20		CAMx Updates	
Threads	**Sec**	**Eff Threads**	**Sec**	**Eff Threads**	**Sec**	**Eff Threads**	**Sec**	**Eff Threads**
1	4781	1.0	4149	1.0	6221	1.0	3460	1.0
4	1530	3.1	1338	3.1	1956	3.2	1132	3.1
6	1223	3.9	1053	3.9	1533	4.1	934	3.7
8	1006	4.8	871	4.8	1259	4.9	846	4.1
10	924	5.2	779	5.3	1137	5.5	732	4.7
12	848	5.6	722	5.7	1042	6.0	661	5.2

3.2 Final Testing

3.2.1 Speed Tests

A final set of model tests were conducted to compare speed and ozone concentration differences resulting from all of the code and compiler modifications. Specifically these modifications included:

- Removal of IEEE compiler flags
- Improvements to EBI convergence criteria
- Streamlining the EBIRATE routine to remove DDM-specific code
- Revised OMP Loop Scheduling and Collapsing
- Improvements to a few instances of mixed-mode math in chemistry and dry deposition

Tests were conducted using the same TCEQ 2012 modeling database employed during Phase 1 of the project. A single day (May 17) was run with the same model configuration as “Run 3” described in Section 2.1.2:

- 3 grids (36, 12, 4 km)
- PiG turned on
- K-theory vertical mixing
- CB6r2h chemistry mechanism
- No Probing Tools

Both modified and un-modified versions of CAMx were compiled using PGF compiler v13.4-0 and Intel (IFORT) compiler v15.0.2.164. Both compilers were installed on an Intel Xeon E5440, and so all compilations were performed on that machine. The standard “O2” compiler optimization was invoked for all compilations. The PGF flag “-Mconcur=nonuma” was removed in all cases.

Both PGF and IFORT builds of unmodified and modified versions of CAMx were run on two different computers to test differences among compilers and hardware, resulting in 8 separate runs. The specifications for the two computers are listed below:

- Dual 12-core Intel Xeon X5660 chipset, 2.8 GHz, 48 GB RAM
- Quad 16-core AMD Opteron 6380 chipset, 2.5 GHz, 128 GB RAM

All runs on both machines utilized 24 cores total, divided among 6 MPI sub-domains by 4 OMP threads, in accordance with scripts we received from TCEQ. Table 3-7 displays run times for the single-day simulations, resulting relative speedup between the unmodified and modified model, and domain-wide maximum surface ozone differences on each of the three grids between the unmodified and modified model.

Table 3-7. CAMx run times for a single simulation day (May 17, 2012) using the original model vs. incorporating all speed updates, and domain-wide maximum ozone differences resulting from the updates. CAMx was built using PGF and IFORT compilers and run on Intel and AMD chipsets.

Chipset	Compiler	Original CAMx (min)	Updated CAMx (min)	Reduction (%)	Max Ozone Difference (ppb)		
					Grid 1	Grid 2	Grid 3
Intel	PGF	185	153	17%	0.157	0.817	0.921
Intel	IFORT	229	124	46%	0.348	3.013	1.114
AMD	PGF	169	135	20%	0.157	0.817	0.921
AMD	IFORT	371	116	69%	0.176	0.817	1.114

Using PGF, model speed was consistently reduced by roughly 20% on the Intel and AMD machines. Much larger improvements of 50% or more were noted for IFORT, particularly on the AMD machine. However, the unmodified IFORT model was much slower on both machines than its PGF counterpart, and so the updates were effective in aligning model speed performance across compilers and hardware. The rather slow performance of the original IFORT model on the AMD machine is concerning, but may be related to the fact that IFORT compilation was executed on an Intel chipset and so not optimized for AMD. We suspect that the IEEE compiler flag for the IFORT/Intel run may have played a role in this feature. Since this should not be an issue for TCEQ's PGF compilations, we plan to further test this issue on our own in the near future.

Maximum concentration differences are generally well below 1 ppb. However, concentration differences up to a few ppb are noted in Table 3-7 (Intel chipset, IFORT build). Differences of this magnitude are most likely related to small changes in the behavior of individual PiG puffs (chemistry, location, size) as a result of the various modifications implemented in CAMx. This result is consistent with the PiG-related impacts identified in the interim testing (Section 3.1.3).

3.2.2 Profiling Results

A final profiling test was run for the updated version of CAMx. The model date and configuration was identical to Phase 1 "Run 3" so that comparisons could be made directly to profiling results from the original version of CAMx (Table 2-4). Table 3-8 presents timing results from OMP thread 0 (consisting of all model processes). The sum of all processes and individual routines listed comprised 93% of the 10,308 second total CPU time for thread 0. Total run time was reduced 14% relative to the original Run 3. Although the order and relative time spent in each model process remained similar to Table 2-4, the updates reduced time spent in OMP and system routine substantially (from 1100 seconds to 553 seconds, or from 9% to 6% of total run time, or about 34% of the total reduction in run time).

Table 3-8. Rerun of Run 3 (Table 2-4) with all compiler flag and code updates. Process and routine CPU time for OMP thread 0. Routines parallelized with OMP are noted.

Process	Routine	OMP	Time (s)	Time (% of total)
Chemistry 45%	ebirate3	●	1,770	17%
	ebisolv	●	839	8%
	hr_hox3	●	648	6%
	ebirxn3	●	461	4%
	chemdriv	●	369	4%
	hr_nox3	●	360	3%
	hr_nxy3	●	150	1%
	hr_pan3	●	129	1%
	kphoto	●	54	<1%
Diffusion 10%	diffuse	●	635	6%
	tridiag	●	317	3%
	vdiffimp	●	123	1%
V Advection 10%	zadvec	●	321	3%
	tridiag	●	317	3%
	vrtslv	●	278	3%
	zrates	Partial	73	<1%
H Advection 10%	hadvppm	●	718	7%
	xyadvec	●	266	3%
PiG 6%	pigdrive	Partial	585	6%
Output 3%	average		154	1%
	massum		153	1%
	aggreg	●	84	<1%
OMP 3%	_mp_barrier		295	3%
System 3%	__c_mzero4		193	2%
	__c_mcopy4		65	<1%
Emissions 1%	Emiss	Partial	135	1%

3.3 Testing at TCEQ

The updated CAMx code was delivered to TCEQ for further testing on the agency's computer system. The specifications of the computer cluster used for these tests are listed below:

- Seven Dell PE M620 nodes with dual 8-core Xeon E5-2650v2 chipsets (112 cores total), 2.6 GHz, 32 GB RAM
- InfiniBand (IB) Network, 40Gb/s
- Dell NFS Storage Solution (NSS; Dell PE R710 server and Dell PV MD1200 disk arrays) – XFS file systems accessed by server nodes via NFS over IB network
- IBM Platform HPC cluster management software

Both modified and un-modified versions of CAMx were compiled using PGF compiler v15.7. For both compilations the standard "O2" compiler optimization was invoked, but the PGF compiler flag "-Mconcur= nonuma" was removed. Additionally, the modified CAMx code was compiled with PGF compiler flag "-Kieee" removed. In both cases, CAMx was parallelized using MPICH v3.1.4 across 6 nodes and with 16 OMP threads on each node, utilizing 96 physical cores without hyper-threading.

Table 3-9 presents the timing results from 2-day tests (June 14 and 15) from TCEQ's June 2012 modeling episode using the same CAMx configuration as described in Section 3.2.1. TCEQ obtained model run times of less than 1 hour per simulation day using CAMx v6.20. Speed improvements of about 15% were realized with the CAMx updates, which is consistent with our results for the PGF compiler on Intel chipsets (Table 3-7). TCEQ found maximum ozone concentration differences similar to the values shown in Table 3-7.

Table 3-10 presents additional timing results obtained by the TCEQ with PiG turned off. Greater speed improvements are realized without PiG, by roughly 8%. For the TCEQ, PiG took roughly 30% of the total model run time, compared to less than 10% in our tests. This may be related to fewer active PiG puffs on the day of our tests (May 17) vs. TCEQ's (June 14-15). More likely, it is because the core model scales better with number of threads than does PiG, combined with TCEQ's use of 96 cores as compared to 24 in our tests. Furthermore, in the TCEQ's tests PiG consistently took ~1000 seconds (17 minutes) for both CAMx v6.20 and the updated model, showing that PiG is insensitive to speed improvements implemented in this project. Speed improvements for PiG should be addressed in future work.

Table 3-9. CAMx runtimes for the unmodified CAMx model and the updated code with all speed improvements: TCEQ's 2-day run with MPI and OMP parallelization over 96 cores.

Episode Day	CAMx v6.20 (s)	Updated CAMx (s)	Reduction (%)
June 14	3332	2773	17%
June 15	2993	2632	12%

Table 3-10. As in Table 3-9, but with PiG turned off. The "Time for PiG" is the absolute runtime consumed by PiG for each day and for each version of CAMx.

Episode Day	CAMx v6.20 (s)	Updated CAMx (s)	Reduction (%)
June 14	2246	1681	25%
Time for PiG	1086 [4]	1092	
June 15	2042	1644	19%
Time for PiG	951	988	

[4] For CAMx v6.20 on June 14, Time for PiG = 3332 s – 2246 s

4.0 CONCLUSION AND RECOMMENDATIONS

Recent additions and updates to CAMx have increased demands on computer resources and have extended model runtimes. TCEQ plans to conduct seasonal modeling with PM and high resolution grids, all of which increase computational demands. The objectives of this project included (1) identifying areas of the CAMx code where improvements would likely have the most impact on model speed, (2) developing and testing various methods to achieve speed improvements, and (3) documenting the speed and accuracy impacts of these modifications using a TCEQ modeling dataset.

Ultimately we implemented several modifications that resulted in the following speed improvements on two of our computer systems using the TCEQ 2012 modeling database with halogen chemistry (CB6r2h) and the Plume-in-Grid (PiG) module:

- 15-30% speed improvement when compiled using PGF compiler;
- 45-50% speed improvement when compiled using IFORT compiler.

The project was conducted under two phases. In Phase 1, we identified areas of the model most needing speed enhancements according to analyses conducted using third-party code profiling tools. Two issues were immediately apparent from our analysis:

- TCEQ should use the "K-theory" vertical diffusion in lieu of the ACM2 option. This is a runtime option that requires no alternative input data or model compilation, and generates practically identical results in much less time. EPA is currently improving the ACM2 solver and this can be brought into CAMx in the future.
- An unnecessary PGF compiler flag "-Mconcur=nonuma" was found to dramatically increase CAMx run times with PGF v13.4. TCEQ should remove this option in case their PGF compiler version is similarly affected by this flag.

Under Phase 1 we found that the chemistry solver, and specifically a single routine that calculates reaction rates, comprises up to 50% of simulation time and that the addition of halogen chemistry slowed the model down disproportionately to the number of additional species and reactions. We also identified certain issues related to our implementation of OMP parallelization in CAMx. Finally, we expect that the current structure of multi-dimensional variables and the order of process splitting (emissions, transport, chemistry) likely have some negative impact on model speed related to memory access (caching) efficiency and overhead associated with MPI parallelization. Addressing these last issues requires greater effort and more fundamental changes to the CAMx code structure, but carries less certainty for speed improvements. We recommend specific activities for future work below.

Phase 2 addressed the highest priorities identified in Phase 1 according to available project resources and schedule. We implemented and tested several modifications to CAMx code, OMP parallelization and compiler flags. The following specific modifications were most effective in improving model speed:

- Remove IEEE compiler flags that standardize mathematical calculations among compilers and thus preclude the use of compiler-specific optimized routines;
- Improve mixed-mode math statements for explicit typing (real, integer, single/double precision, etc.) to maintain accuracy without the need for IEEE flags;
- Tailor OMP loop scheduling and collapsing to maximize OMP efficiency and scalability;
- Update chemistry solver convergence criteria; this strategy is particularly effective for halogen chemistry;
- Remove DDM-specific calculations from the reaction rate routine (we have updated the Chemical Mechanism Compiler to automate this change across all mechanisms).

We found that halogen chemistry derives more speedup than the standard CB6r2 mechanism, and that the modifications tend to align model speed performance across compilers and hardware. Impacts to peak ozone concentrations from these modifications were found to be well below 1 ppb in cases without PiG, to generally just below 1 ppb (and in some cases just exceeding 1 ppb) with PiG. This is attributed to small changes in PiG behavior that cause puffs to dump mass to the grid at slightly different times (and possibly locations). We see no impacts to OMP scalability with these speed improvements.

We have provided the updated version of CAMx to TCEQ at the close of this project. Tests were performed on the agency's computer cluster system using the same June 2012 datasets and model configuration as our tests. TCEQ's utilization of 96 cores for parallelization results in model run times of less than 1 hour per simulation day. Additional speed improvements of about 15% are realized with the CAMx updates, which are consistent with our results. Greater speed improvements by roughly 8% are realized without PiG. Removal of PiG in TCEQ's tests results in a consistent ~1000 second (17 minute) improvement in both the original model and the updated model, suggesting that PiG is insensitive to speed improvements implemented in this project.

CAMx code updates described here will be incorporated into the next public release of CAMx.

4.1 Recommendations

We recommend that TCEQ conduct tests over their longer modeling periods to benchmark speed improvements. We also suggest that TCEQ apply CAMx with a variety of OMP/MPI combinations on their computer system to see if these updates impact optimum parallelization.

Based on results from this project, we recommend follow-on work to further improve CAMx speed performance:

- Update the ACM2 vertical diffusion solver according to EPA's improvement for CMAQ;
- Screen for instances where mixed-mode math statements are inadequately coded and make appropriate adjustments;
- Implement efficiency and OMP parallelization improvements in the Plume-in-Grid model;

- Add OMP parallelization to currently un-parallelized portions of the code;
- Further investigate and test efficiency from scheduling OMP loops as static or dynamic;
- Investigate and test improvements to MPI parallelization for processes that consistently and inherently suffer from load imbalance, e.g., Plume-in-Grid;
- Restructure major 3- and 4-D variable arrays (in combination with the point below) for more efficient memory management and caching;
- Restructure the order of key physical processes in the model (emissions, transport, chemistry, PiG) to operate together more efficiently by improving memory caching and reducing MPI overhead.

5.0 REFERENCES

Emery, C., B. Koo, T. Sakulyanontvittaya, G. Yarwood, 2013. Improving CAMx GREASD PiG Efficiency. WO 582-11-10365-FY13-09 Final Report. Prepared for the Texas Commission on Environmental Quality, Austin, TX. Prepared by ENVIRON International Corp., Novato, CA (August, 2013).

Emery, C., G. Wilson, D.J. Rasmussen, G. Yarwood, 2015. Work Order No. 582-15-52944-FY15-35, Phase 1: CAMx Speed Improvement Strategy Plan. Memorandum prepared for Jim MacKay, Texas Commission on Environmental Quality, Austin, TX. Prepared by ENVIRON International Corp., Novato, CA (March 31, 2015).

ENVIRON, 2012. Dallas-Fort Worth Modeling Support: Improving Vertical Mixing, Plume-in-Grid, and Photolysis Rates in CAMx. WO 582-11-10365-FY12-06 Final report. Prepared for the Texas Commission on Environmental Quality, Austin, TX. Prepared by ENVIRON International Corp., Novato, CA (August, 2012).

ENVIRON, 2015. The Comprehensive Air quality Model with extensions (CAMx), version 6.20. http://www.camx.com.

Johnson, J., E. Tai, P. Karamchandani, G. Wilson, G. Yarwood, 2013. TCEQ Ozone Forecasting System. WO 582-11-10365-FY13-13 Final Report. Prepared for the Texas Commission on Environmental Quality, Austin, TX. Prepared by ENVIRON International Corp., Novato, CA (November 15, 2013).

Johnson, J., G. Wilson, DJ Rasmussen, G. Yarwood, 2015. Daily Near Real-Time Ozone Modeling for Texas. WO 582-11-10365-FY14-16 Final Report. Prepared for the Texas Commission on Environmental Quality, Austin, TX. Prepared by ENVIRON International Corp., Novato, CA (January 29, 2015).

Wilson, G. and J. Johnson, 2010. Speed Enhancements for EPS3. WO 582-7-84005-FY10-09 Final Report. Prepared for the Texas Commission on Environmental Quality, Austin, TX. Prepared by ENVIRON International Corp., Novato, CA (July, 2010).