PERFORMANCE ANALYSIS OF CAMx ON COMMODITY PLATFORMS

George Delic*
HiPERiSM Consulting, LLC, Durham, NC

Byeong-Uk Kim and Harvey E. Jeffries
Department of Environmental Sciences and Engineering, University of North Carolina, Chapel Hill, NC

1. INTRODUCTION

This is a progress report on a project to evaluate industry standard fortran 90/95 compilers for IA-32 Linux™ commodity platforms when applied to Air Quality Models (AQM). The goal is to determine the optimal performance and workload though-put achievable with commodity hardware for such models because they are in wide-spread use on these platforms. New results are presented for CAMx 4.03 that give insight into the algorithm's performance on commodity architectures. Important performance bottle-necks are identified with the aid of proprietary software to collect and compute performance metrics using a publicly available hardware performance interface.

2. CHOICE OF HARDWARE, OPERATING SYSTEM, AND COMPILERS

The hardware used for the results reported here is the Intel Pentium 4 Xeon (P4) and Pentium Xeon 64EMT (P4emt) processors. These have processor clock rates of 3GHz and 3.4GHz, respectively. Each is in a dual configuration with a corresponding front side bus (FSB) of 533MHz and 800HMz shared by each pair of processors. The operating system (OS) is HiPERiSM Consulting, LLC's modification of the Linux™ 2.6.9 kernel to include a patch that enables access to hardware performance counters. This modification allows the use of the Performance Application Programming Interface (PAPI) performance event library (PAPI, 2005) to collect hardware performance counter values as the code executes. The compilers used were the Portland pgf90/95 (release 6.0) and Intel ifort (release 9.0) for the three groups of optimization switches shown in Table 1.

These architectures offer Streaming Single-Instruction-Multiple-Data Extensions, (SSE) to enable vectorization of loops operating on multiple elements in a data set with a single operation. This is enabled through a compiler switch (sse in Table 1) and has been used in these tests.

TABLE 1. P4 compiler command and switches

Compiler and platform	Compiler optimization switches	Switch group mnemonic
pgf95	–O0	noopt
(P4*)	–O2	opt
(P4emt**)	–fast –Mvect=sse	sse
Ifort	-O0 -Ob0 -unroll0	noopt
(P4*)	-O3 -Ob2 -prefetch-	opt
(P4emt**)	-xW -O3 -Ob0 -prefetch-	sse

* Other P4 options include (a) for pgf95: –tp p7 -pc 64 -Mdalign -Mextend -Mnoframe -Mlfs -byteswapio –Wl, -Bstatic, and (b) for ifort: -tpp7 -FI -convert big_endian

** Other P4emt options include (a) for pgf95: -tp p7-64 -mcmodel=medium -Mextend -Mnoframe –byteswapio, and (b) for ifort: -tpp7 -FI -convert big_endian -mcmodel=large -i_dynamic

3. CHOICE OF BENCHMARKS

The CAMx code developed by ENVIRON (ENVIRON) is a Fortran 77 code for an Eulerian photochemical model that is widely used in the AQM community. This benchmark analysis includes a selected case for the 2000 episode on 8/22 in the Houston Greater Metro area, that is labeled as "base5a.regular" by the modeling team at the Texas Commission on Environmental Quality (TCEQ). Modeling files are obtainable from the TCEQ URL (TCEQ)

[*] *Corresponding author:* George Delic, HiPERiSM Consulting, LLC, P.O. Box 569, Chapel Hill, NC 27514-0569; e-mail: george@hiperism.com

4. HARDWARE PERFORMANCE EVENTS

The PAPI (PAPI, 2005) interface defines over a hundred hardware performance events, but not all of these events are available on all platforms. For the Intel hardware under discussion the number of hardware events that can be collected are, respectively, 28 (P4) and 25 (P4emt) and Table 2 only events that are common to them. Not all events can be collected in a single execution due to the fact that the number of hardware counters is small (typically four). Thus, multiple executions are needed to collect all available events on any given platform. The process time (PTIME) reported here is obtained from the hardware performance counter interface.

TABLE 2. PAPI events common to the Intel P3, P4 and P4emt.

Category	Description	Name
Floating Point Operations	Floating point instructions	PAPI_FP_INS
	Floating point operations	PAPI_FP_OPS
Instruction Counting	Total cycles	PAPI_TOT_CYC
	Instructions issued	PAPI_TOT_IIS
	Instructions completed	PAPI_TOT_INS
	Vector/SIMD instructions	PAPI_VEC_INS
Data Access	Cycles stalled on any resource	PAPI_RES_STL
Cache Access	L1 data cache misses	PAPI_L1_DCM
	L1 load misses	PAPI_L1_LDM
	L1 instruction cache accesses	PAPI_L1_ICA
	L1 instruction cache misses	PAPI_L1_ICM
	L2 load misses	PAPI_L2_LDM
	L2 store misses	PAPI_L2_STM
	L2 total cache misses	PAPI_L2_TCM
TLB Operations	Data translation lookaside buffer misses	PAPI_TLB_DM

5. PERFORMANCE METRICS

5.1 Rate performance metrics

Rate metrics have the suffix "_rate" (except for MFLOPS) and some examples include TOT_CYC_rate, TOT_INS_rate, L1_DCM_rate, and L2_TCM_rate. This naming convention uses the corresponding PAPI event name in Table 2 divided by the process time with units of million per second. The following discussion will use those rate metrics of relevance in identifying bottle-necks in CAMx.

TABLE 3. Examples of ratio metrics common to the Intel P4 and P4emt.

Description	Name
Memory instructions versus total instructions	MEM_INS_TOT
Memory instructions per floating point instruction	MEM_INS_FPINS
Data TLB misses per floating point instruction	TLB_DM_FPINS
Respectively, L1 instruction, data, and total cache misses per floating point instruction	L1_ICM_FPINS L1_DCM_FPINS L1_TCM_FPINS
L2 total cache misses per floating point instruction	L2_TCM_FPINS

5.2 Ratio performance metrics

In addition to rate metrics, ratios of PAPI events define a set of ratio metrics. Table 3 lists a few examples of ratio metrics used in the following discussion to identify performance bottle-necks in CAMx. Other rate metrics are introduced as needed.

5.3 Profiling and code performance

While not a metric, execution profiling is useful in determining where "hot spots" occur in the source code by measuring (cumulative) time consumed during the code execution. A profile of CAMx is discussed to identify the compute intensive routines and their code characteristics.

6. CAMx PERFORMANCE RESULTS

6.1 Operations, instructions, and cycles

Fig. 1 shows the process time for CAMx on P4 and P4emt platforms. The left and right hand half of Fig. 1 shows, respectively, the P4 and P4emt results. Each group of executions corresponds to the same choice of compiler switches listed in Table 1. Comparing the pgf90 and ifort results on the P4 platforms shows the shortest times are for the pgf-opt and ifc-sse cases with 11,771and 12,823 seconds, respectively. Surprisingly, the P4emt results are not an improvement on the P4 values. This is despite the fact that the TOT_CYC_rate (cycles rate) and TOT_INS_rate (instruction rate) show the increase expected from the change in clock rate between 3GHz and 3.4GHz.

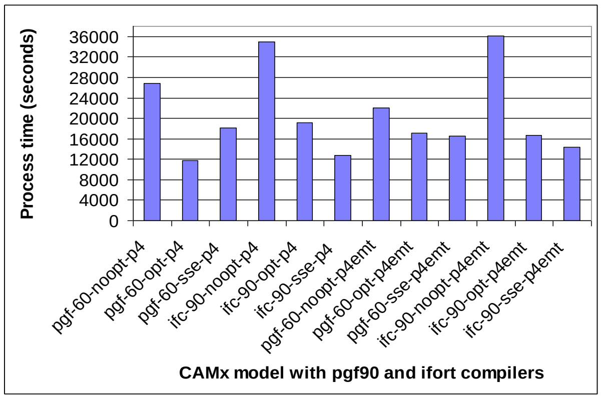

Fig. 1. Process time for CAMx with (alternately) pgf90 and ifort compilers on P4 and P4emt processors (left and right half, respectively). Each compiler has the three groups of compiler switches defined in Table 1.

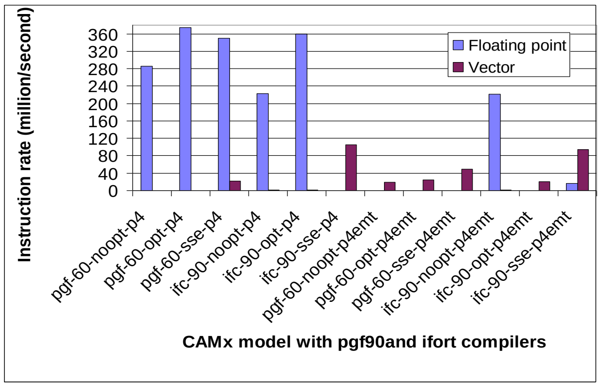

Fig. 2. Arithmetic instruction rates for CAMx with (alternately) pgf90 and ifort compilers on P4 and P4emt processors (left and right half, respectively). Each compiler has the three groups of compiler switches defined in Table 1.

The higher level optimizations (sse) give little performance gain on the P4emt. When sse switches were enabled for the P4 the pgf90 time increased whereas the ifort time decreased. This confusing behavior is related to the way in which each compiler uses the sse instruction set on the two platforms. Fig. 2 shows the floating point (fp) and vector instruction rates corresponding to the optimization switch groups in Table 1. On the P4 the pgf90 (sse) results show negligibly small vector instruction rates compared to fp whereas the reverse is true for ifc (sse). On the P4emt the fp instruction counts reported by PAPI have all but disappeared (with one exception) and vector instruction rates dominate. Thus, it appears that when either compiler detects the 64 bit hardware on a 64 bit kernel, it attempts to use the enhanced sse instruction set (with the exception of ifc-noopt). This approach takes advantage of the availability of considerably more hardware resources on the P4emt compared to the P4 when operating with a 64 bit Linux kernel. However, one side effect when vector instructions predominate is that the Mflops reported by the PAPI event counter PAPI_MFLOPS underestimate Mflops. This is because they are based on the fp operation count. For those cases where such Mflops are correctly estimated they are shown in Fig. 3. The Mflops range from a low of 225 (ifc-noopt) to 375 (pgf-opt).

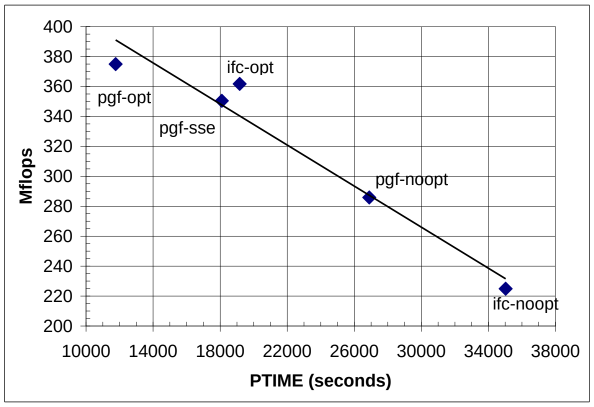

Fig. 3. Floating point rates in million per second (Mflops) for CAMx with pgf90 (pgf) and ifort (ifc) compilers on the P4 processor. Each compiler has one of the three groups of compiler switches defined in Table 1.

6.2 Memory footprint

In comparing performance of compilers and processors the memory behavior is of special interest. Fig. 4 shows instruction rates for load (LD_INS_rate), store (SR_INS_rate), and the sum of the two (MEM_TOT_rate). This shows that enabling optimization (with either compiler) reduces memory instruction rates. Also there is a small tendency to reduce the memory instruction rate when the same compiler is compared on the two platforms. In general, Fig. 4 shows that the rate of total memory instructions issued (loads plus stores) is voluminous. A high rate of memory instruction issue need not be an indicator of a performance bottleneck. Benchmarks with good vector character that deliver of the order of 1Gflop on a P4 can also show high memory access rates. However, an interesting differentiator is the number of memory instructions issued per floating point instruction.

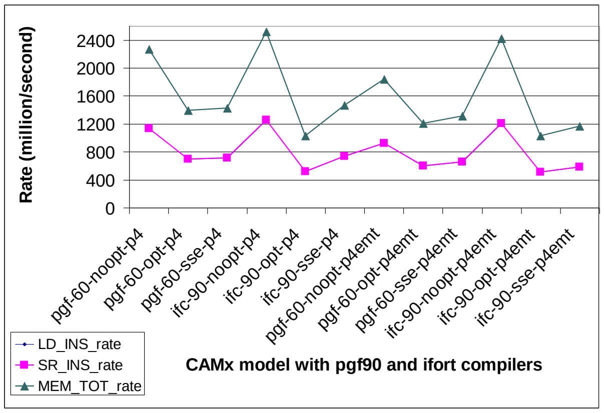

Fig. 4. Memory instructions in million per second for CAMx with (alternately) pgf90 and ifort compilers on P3 and P4emt processors (left and right half, respectively). Each compiler has the three groups of compiler switches defined in Table 1.

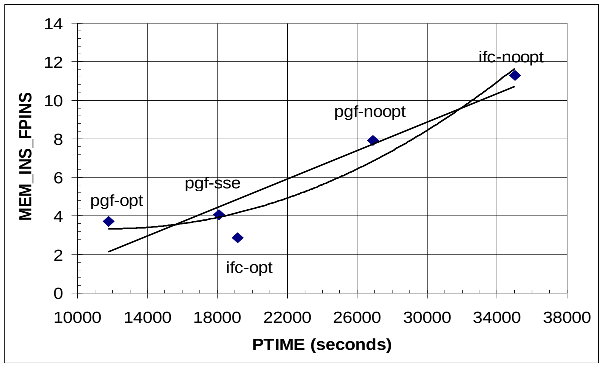

Fig. 5. Number of memory instructions per floating point instruction versus process time for CAMx on the P4 processor with pgf90 (noopt, opt, sse) and ifort (noopt, opt) executions. Regression lines are added to show that process time increased with increasing MEM_INS_FPINS.

Fig. 5 shows the correlation between this metric (MEM_INS_FPINS) and the process time (PTIME). There is a simple correlation and both linear and quadratic regression lines are shown. Lower values MEM_INS_FPINS correspond to smaller process time. This shows that process time is longest when memory instructions per fp instruction are as large as 7.9 (pgf-noopt) and 11.3 (ifc-noopt) when optimization is disabled. Conversely, the range 2.9 (ifc-opt) to 4 (pgf-sse), or 3.7 (pgf-opt) shows the lower values of PTIME with the smallest time for the pgf-opt case.

The results of the MEM_INS_FPINS metric suggests that CAMx is a memory-intensive algorithm. However, a memory intensive application, without a dominant vector code character (as is CAMx), is performance constricted on commodity architectures where memory bandwidth is limited by the FSB and cache design. The consequence of CAMx's memory footprint is that cache can become a limiting critical resource and this is explored in the next section.

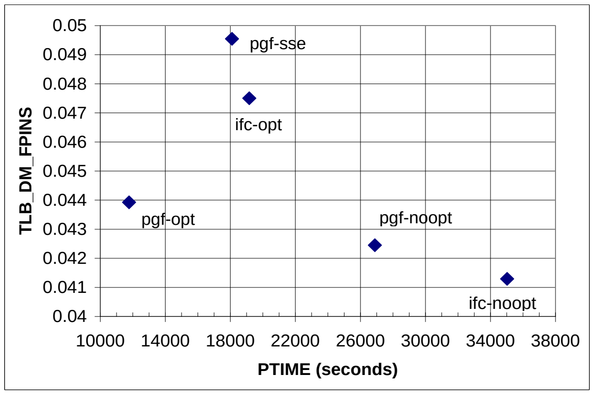

Fig. 6. Number of data TLB cache misses per floating point instruction versus process time for CAMx on the P4 processor with pgf90 (noopt, opt, sse) and ifort (noopt, opt) executions.

Between the processor and the first level of cache (L1) there is the TLB cache. The translation lookaside buffer (TLB) is a small buffer (or cache) to which the processor presents a virtual memory address and looks up a table for a translation to a physical memory address. If the address is found in the TLB table then there is a hit (no translation is computed) and the processor continues. The TLB buffer is usually small, and efficiency depends on hit rates as high as 98%. If the translation is not found (a TLB miss) then several cycles are lost while the physical address is translated. Therefore TLB misses degrade performance. PAPI offers counters for TLB miss events for both instruction and data. In the case of CAMx it is the data TLB misses that are critical. Fig. 6 shows the data TLB misses per fp instruction (TLB_DM_FPINS) versus process time (PTIME). From this graph it is clear that the execution with the shortest time (pgf-opt) has the lowest number of dataTLB cache misses in the group of three points with the lowest process times in Figs. 5 and 6. However, a complete explanation of CAMx behavior on the P4 platform is more subtle, and depends on cache performance.

6.3 Cache usage

Both the P4 and P4emt platforms discussed here have L1 and L2 caches. A cache miss occurs when data or instructions are not found in the cache and an excursion to higher level cache, or memory, is necessitated. Cache misses result in lost performance because of increasing latency in the memory hierarchy. Memory latency is smallest at the register level and increases by an order of magnitude for a L1 cache reference, and another order of magnitude to access L2 cache.

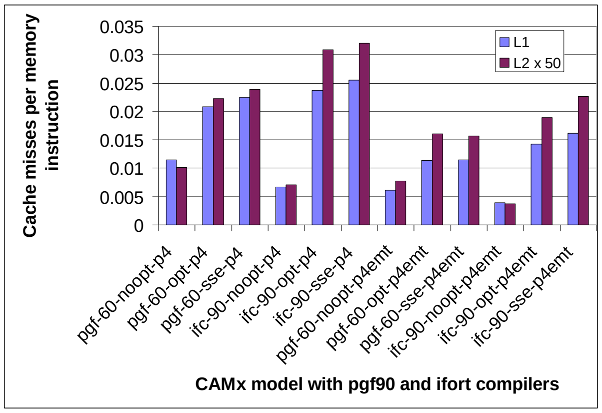

Fig. 7. L1 and L2 cache misses per memory instruction in million per second for CAMx with (alternately) pgf90 and ifort compilers on P3 and P4emt processors (left and right half, respectively). Each compiler has the three groups of compiler switches defined in Table 1.

In the case of CAMX the number of cache misses per memory instruction is shown in Fig. 7 for P4 and P4emt platforms. The L1 results are the data cache misses whereas for the L2 they are the total cache misses (in both cases the instruction cache misses are negligible by comparison). The L2 values have be scaled by a factor of 50. The key here is that while the pgf90-noopt case appears with low values, it has more than twice the memory references than the pgf-opt case.

There is another view of the penalties associated with excursions to cache by the processor. Figs. 8 and 9, respectively, show rates for data TLB misses versus the L1 and L2 cache misses for the P4 and P4emt platforms. It is obvious that increasing data TLB miss rates also results in increasing L1 and L2 data cache miss rates. The range of 10-25 million/second data TBL miss rate is a very large value. When coupled with the correlated L1 and L2 cache misses one important performance bottle-neck in CAMx is clear: voluminous non-sequential data access.

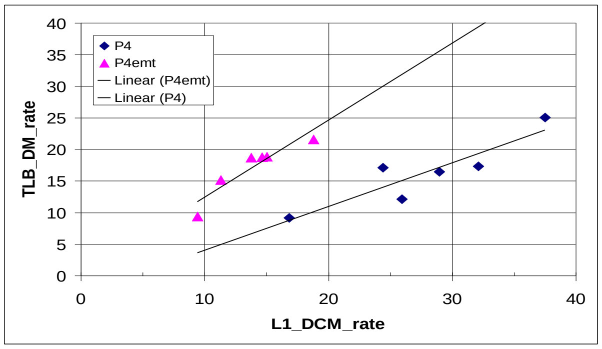

Fig. 8. Data TLB data cache miss rates versus L1 data cache miss rates (both in million per second) for CAMx on the P4 (lower set) and P4emt (upper set). A linear regression line is shown for each case.

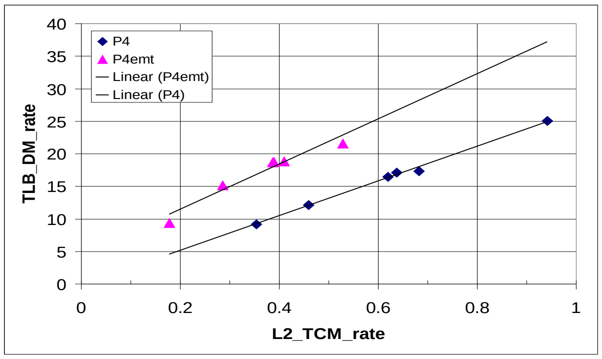

Fig. 9. Data TLB data cache miss rates versus L2 data cache miss rates (both in million per second) for CAMx on the P4 (lower set) and P4emt (upper set). A linear regression line is shown for each case.

Inspection of Figs. 8 and 9, at a fixed value of data TLB misses, say 15 million/second, shows that the P4emt cache miss rates are almost a factor of two less than the P4 results. If cache misses were the critical performance bottle-neck then the P4emt performance of CAMx should be better than the P4 performance. Comparison in Fig. 1 of pgf-sse on the P4 and P4emt shows that there is a performance gain. Conversely, the same comparison for ifc-sse shows a performance degradation. However, neither of these optimizations produce the short process time of pgf-opt on the P4. This suggests that the extremely high data TLB misses for CAMx are a critical source of performance limitations.

7. CAMX EXECUTION PROFILE

An execution profile of CAMx is easily performed with the –Mprof=lines compiler switch in the pgf90/95 compiler. Results with the opt optimization switches is shown in Table 4. This shows those functions accounting for 95% of the cumulative process time with some that have a high calling overhead. Once the important functions are identified code inspection shows some reasons why vector instructions are scarce in CAMx.

TABLE 4. CAMx P4 profile for opt switches

Function	Number of calls	Time (%)
hadvppm	372,106,488	18
cpivot	0	14
radslvr3	1,012,218,877	14
trap	506,467,872	8
diffus	4,252	8
vrtslv	543,220,800	5
trdiag	1,086,441,600	5
ratejac3	1,012,218,877	4
chemdrive	4,252	4
average	8,504	3
xyadvec	4,252	3
ktherm	507,573,254	3
vdiffimp	543,220,800	2
zadvec	4,252	2
massum	10,748	2
		…….
		95

The top three routines account for 46% of the process time but inhibit loop vectorization either because of conditions such as non-vectorizable recurrences or wrongly ordered loop nests. Examples of the latter in radslvr3 clearly lead to data TLB misses when the innermost loop range is on the outermost array index. Subroutines radslvr3, trdiag, and ratejac3 have over 1 million calls, or some 860 calls per second. Others such as trap, vrtslv, ktherm, and vdiffimp have over 0.5 million calls, or some 430 calls per second. Of this list those that account for a negligible amount of the process time but have a very high calling overhead should be inlined to reduce the cost of control transfer instructions. Inline methods are well documented in the PGI User's Guide. Likewise interprocedural optimizations should be applied with the –Mipa compiler switch. Both of these simple steps should give significant reductions of process time and will be tested at a later date. However, the removal of vectorization inhibitors in the top routines would require source code modifications. However, the tridiagonal solver trdiag does have a non-vectorizable recurrence.

8. CONCLUSIONS

This performance analysis of CAMx, shows that this is a memory intensive application with a minimum of 3 to 4 memory operations to each floating point operation. Also, a large value of the data TLB miss rate was measured. In combination these two characteristics of the CAMx code place a limit on the optimal performance possible from CAMx on commodity platforms. This is because, by design, commodity hardware solutions offer a cost effective compromise between processor clock rates, cache size, and bandwidth (or latency) to memory.

In its present form, CAMx gains mostly from improvements in the scalar performance of the hardware. But, despite these observations, a profile of CAMx performance, followed by code inspection, does suggest that there is scope for performance improvement beyond the 375 Mflop range it currently delivers on the P4. Although longer run times were found on the P4emt, this 64 bit platform with a 64 bit kernel more easily supports large files and has better arithmetic precision. The question of I/O performance on the P4 and P4emt platforms is still an open issue.

REFERENCES

ENVIRON: http://www.camx.com

PAPI, 2005: *Performance Application Programming Interface*, http://icl.cs.utk.edu/papi. Note that the use of PAPI requires a Linux kernel patch (as described in the distribution).

TCEQ:
http://www.tnrcc.state.tx.us/air/aqp/airquality_photomod.html#section4
http://www.tnrcc.state.tx.us/air/aqp/airquality_photomod.html#camx