HiPERiSM Consulting, LLC.

George Delic , Ph.D.
HiPERiSM Consulting, LLC
(919)484-9803
P.O. Box 569,
Chapel Hill, NC 27514
george@hiperism.com
http://www.hiperism.com

Models-3 User's Conference
September 26-28, 2005
UNC-Chapel Hill, NC

PERFORMANCE METRICS FOR AIR QUALITY MODELS ON COMMODITY PLATFORMS

George Delic, HiPERiSM Consulting, LLC

Overview

1. Introduction
2. Choice of Hardware, Operating System, & Compilers
3. Choice of Benchmarks
4. Hardware Performance Events
5. Performance Metrics
6. CMAQ Performance Results
7. CMAQ Execution Profile
8. Conclusions

1. Introduction

- Motivation
  - Study CMAQ 4.4 performance
  - Measure performance on Linux platforms
  - Identify performance bottle-necks
  - Evaluate Fortran 90/95 compilers
- Performance tools
  - HiPERiSM's proprietary software interface
    - Use PAPI performance library (*Performance Application Programming Interface*, http://icl.cs.utk.edu/papi)
    - Automatically collect hardware performance events
    - Calculate performance metrics

2. Choice of Hardware, Operating System, and Compilers

- Hardware and OS
  - P3: Intel Pentium 3 (0.933 GHz,133 MHz FSB), Linux 2.4.20 kernel
  - P4: Intel Pentium 4 Xeon (3.0 GHz, 533 MHz FSB), Linux 2.6.9 kernel
  - Linux kernels modified at HC
- Fortran compilers for IA-32 Linux
  - Portland pgf90 4.0 (P3)
  - Portland pgf90/95 6.0 (P4)

2. Choice of Hardware, Operating System, and Compilers (cont.)

Compiler and switch mnemonic	Compiler optimization switches*
pgf-4.0(6.0)-noopt	-O0 –tp p6(p7)
pgf-4.0(6.0)-opt	–O2 –tp p6(p7)
pgf-4.0(6.0)-vect	–fast –Mvect –tp p6(p7)
pgf-4.0(6.0)-sse	–fast –Mvect=sse –tp p6(p7)

* *Corresponding options for the P4 target architecture are shown in parentheses.*

3. Choice of Benchmarks

CMAQ 4.4

- Developed by U.S. EPA
- Available from http://www.cmascenter.org
- Case 1: tutorial day 1 at 32km resolution
- Case 3: tutorial day 1 at 8km resolution

(Cases 2 and 4 are for day 2 and are not included here)

4. Hardware Performance Events

Category	Description	PAPI Name
FP 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

4. Hardware Performance Events (cont.)

Category	Description	PAPI Name
Cache Access (L1)	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
Cache Access (L2)	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 (Rate and Ratio metrics)

Rate metrics

- Root is PAPI name with suffix “ _rate”, e.g.
  - L1_DCM_rate = L1 data cache miss rate (million/second)
  - MEM_TOT_rate = Total memory instruction* rate (million/second)

Exception: Mflops = million floating point operations per second

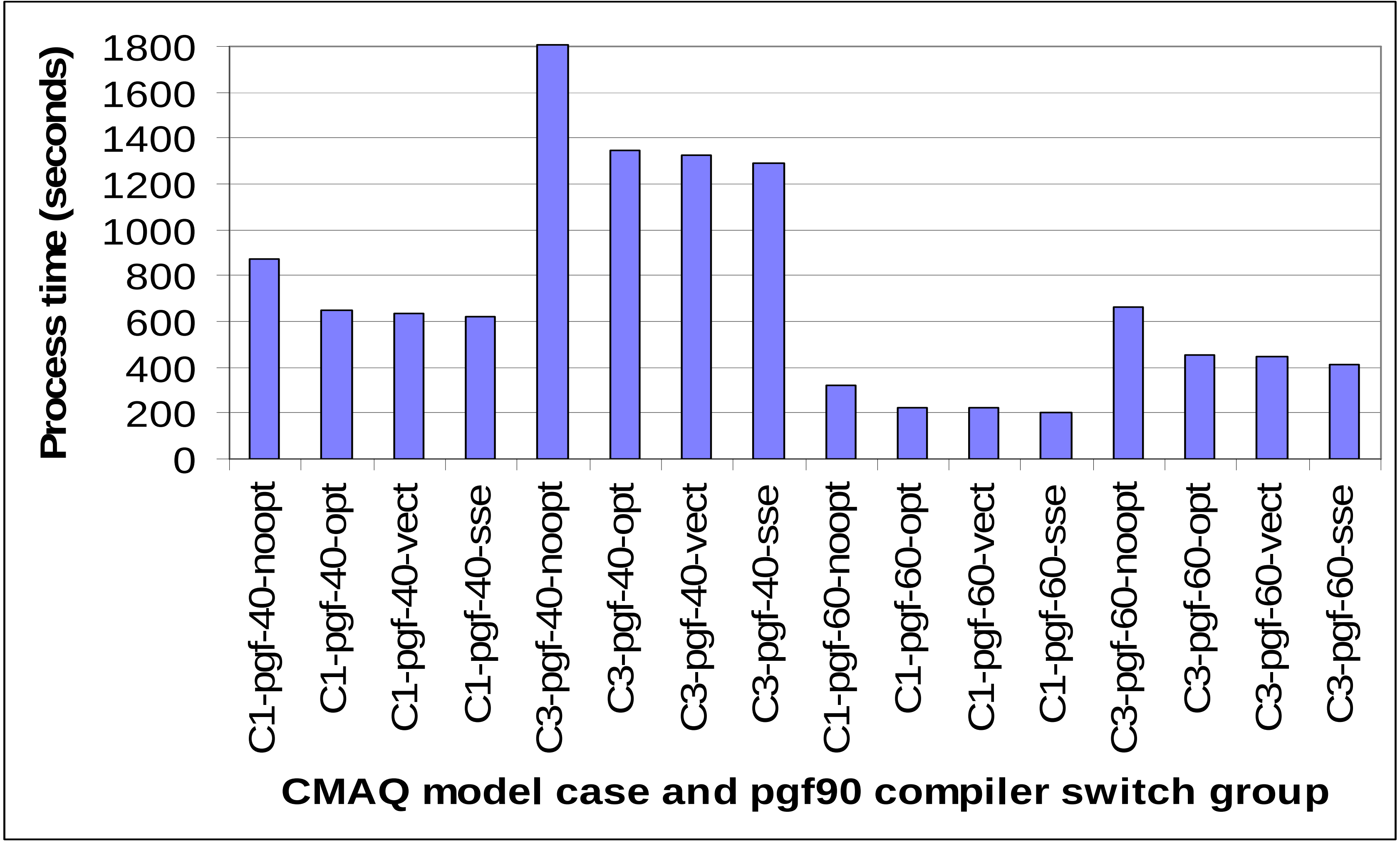
* *MEM_TOT_INS is sum of load (PAPI_LD_INS) and store (PAPI_SR_INS) instructions*

5. Performance Metrics (cont.)

Ratio Metric Definition*	Name
Memory instructions per fp instruction	MEM_INS_FPINS
Data TLB misses per fp instruction	TLB_DM_FPINS
L1 instruction cache misses per flop	L1_ICM_FPOP
L1 data cache misses per flop	L1_DCM_FPOP
L1 total cache misses per flop	L1_TCM_FPOP

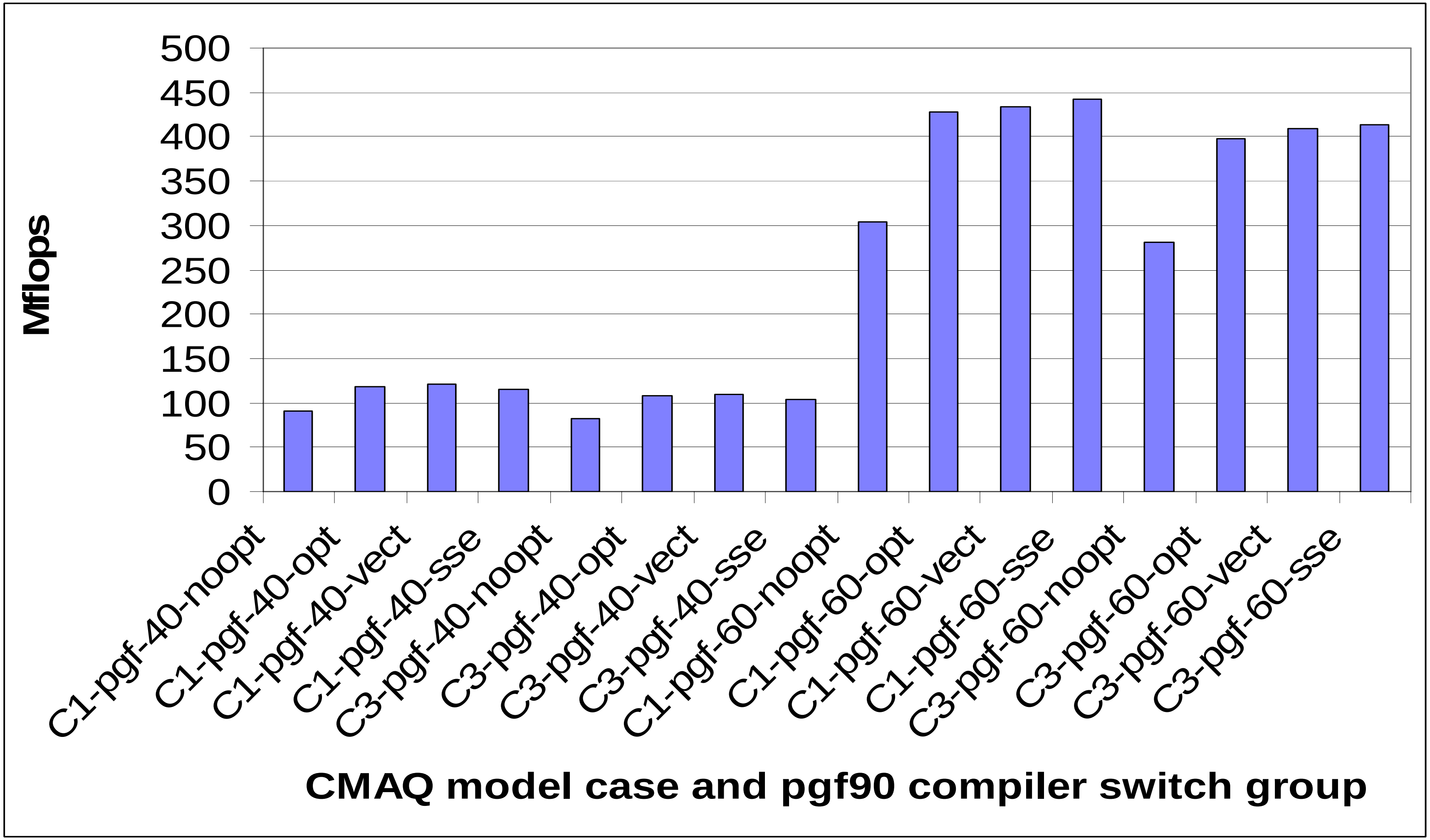
** “fp” denotes floating point, and “flop” denotes fp operation*

6. CMAQ Performance Results (P3 and P4 Process Time)



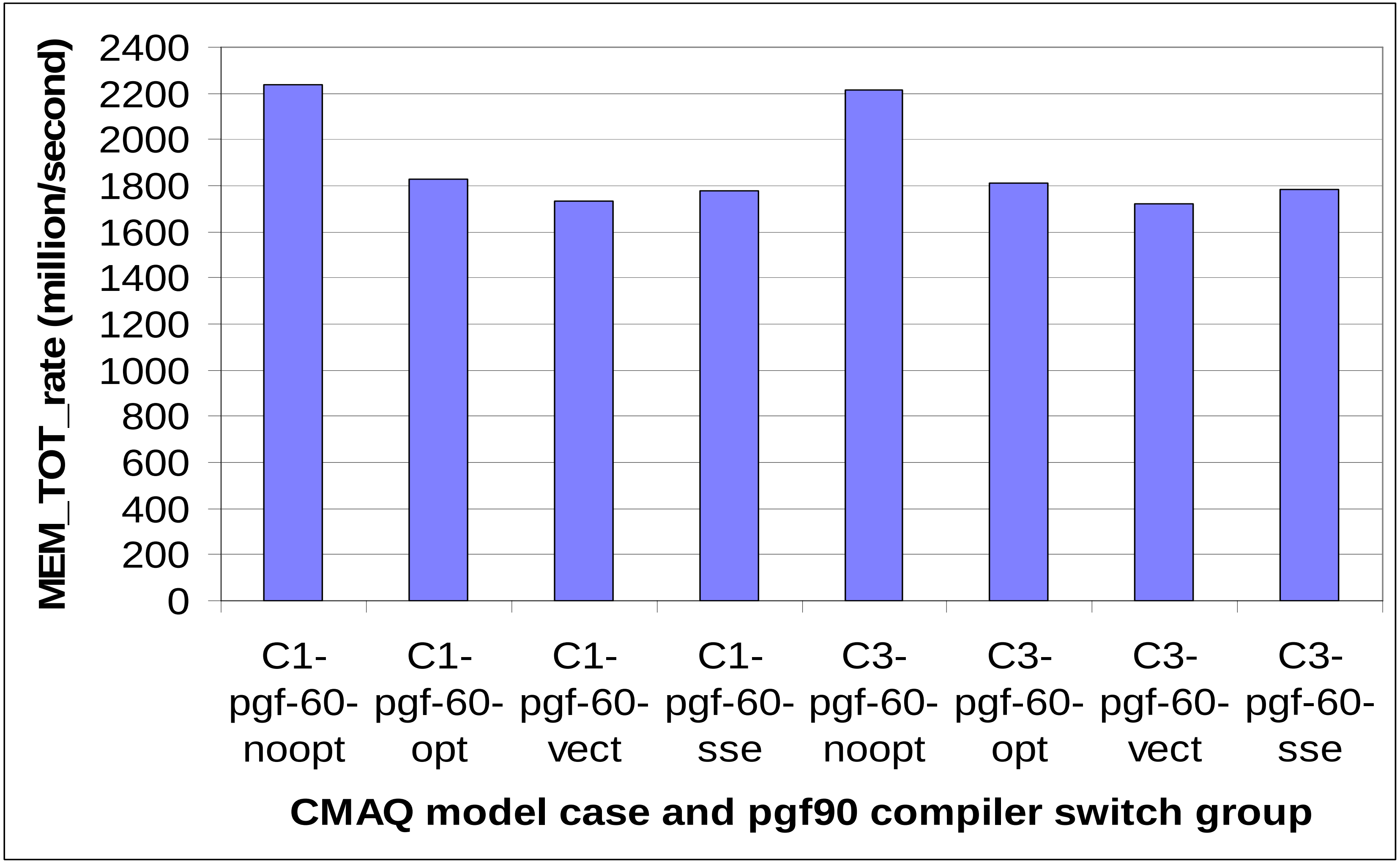
http://www.hiperism.com

6. CMAQ Performance Results (P3 and P4 Mflops)

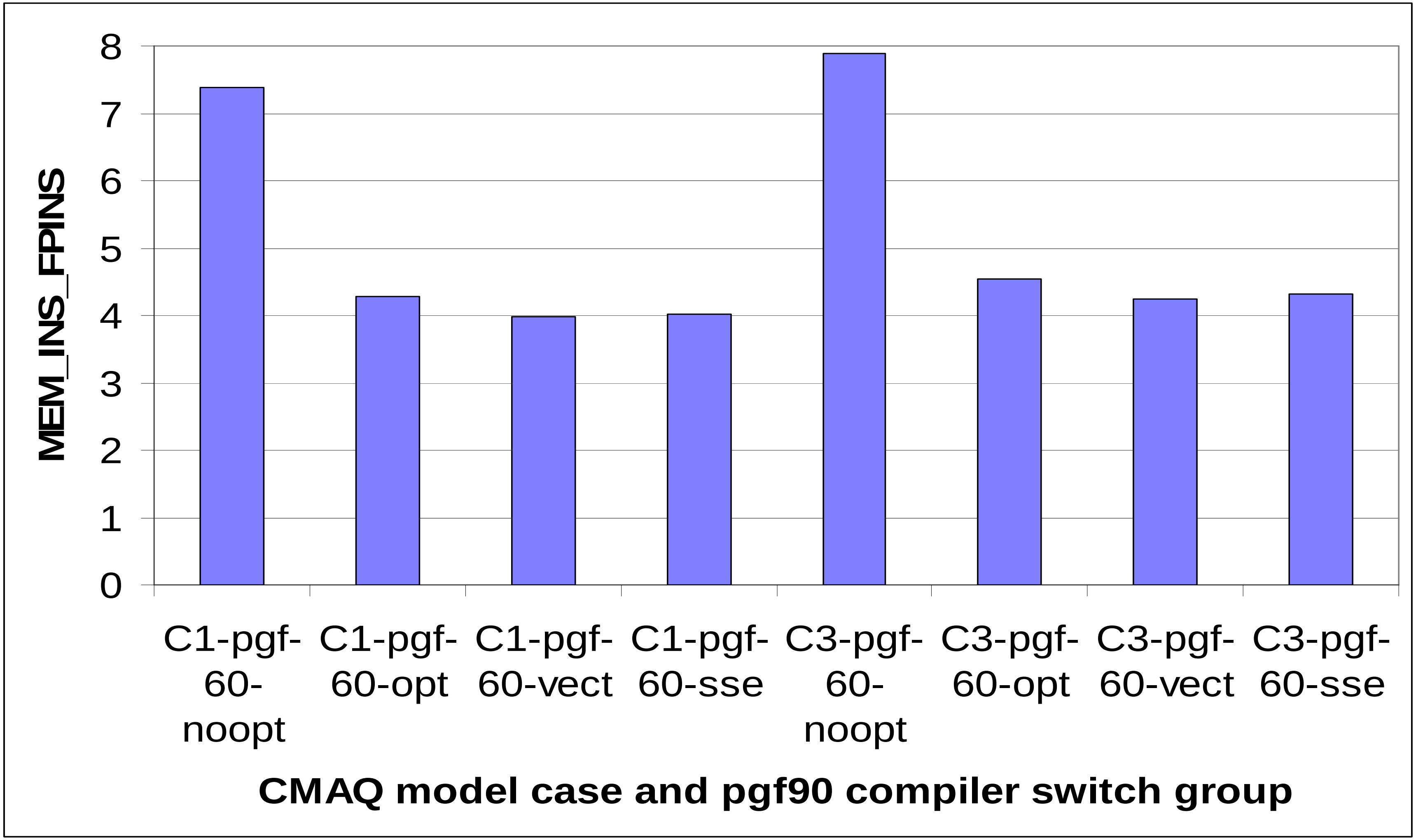


http://www.hiperism.com

6.0 CMAQ Performance Results (Total Memory Inst. rate)

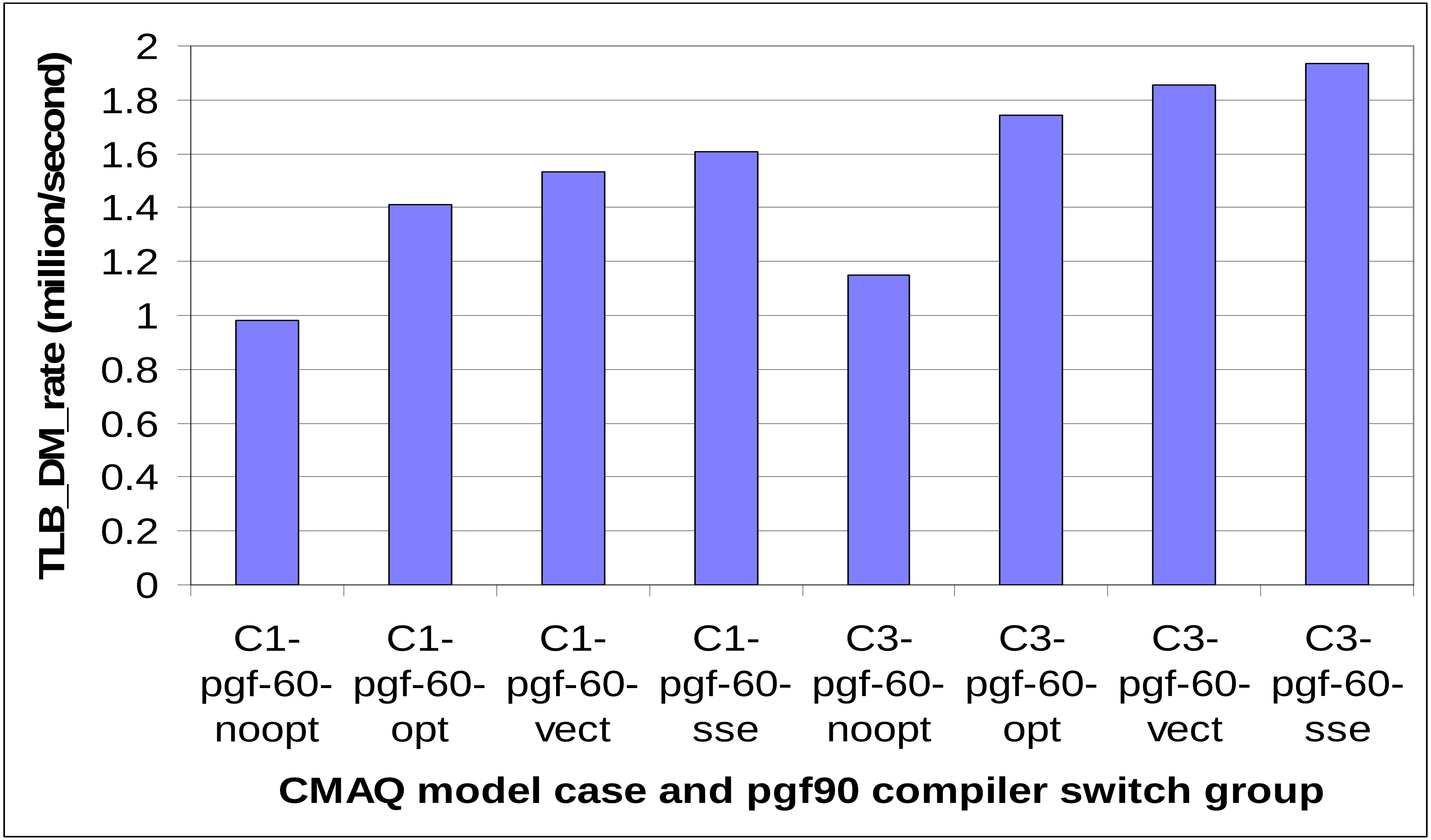


6.0 CMAQ Performance Results (Memory Inst. per FP Inst.)



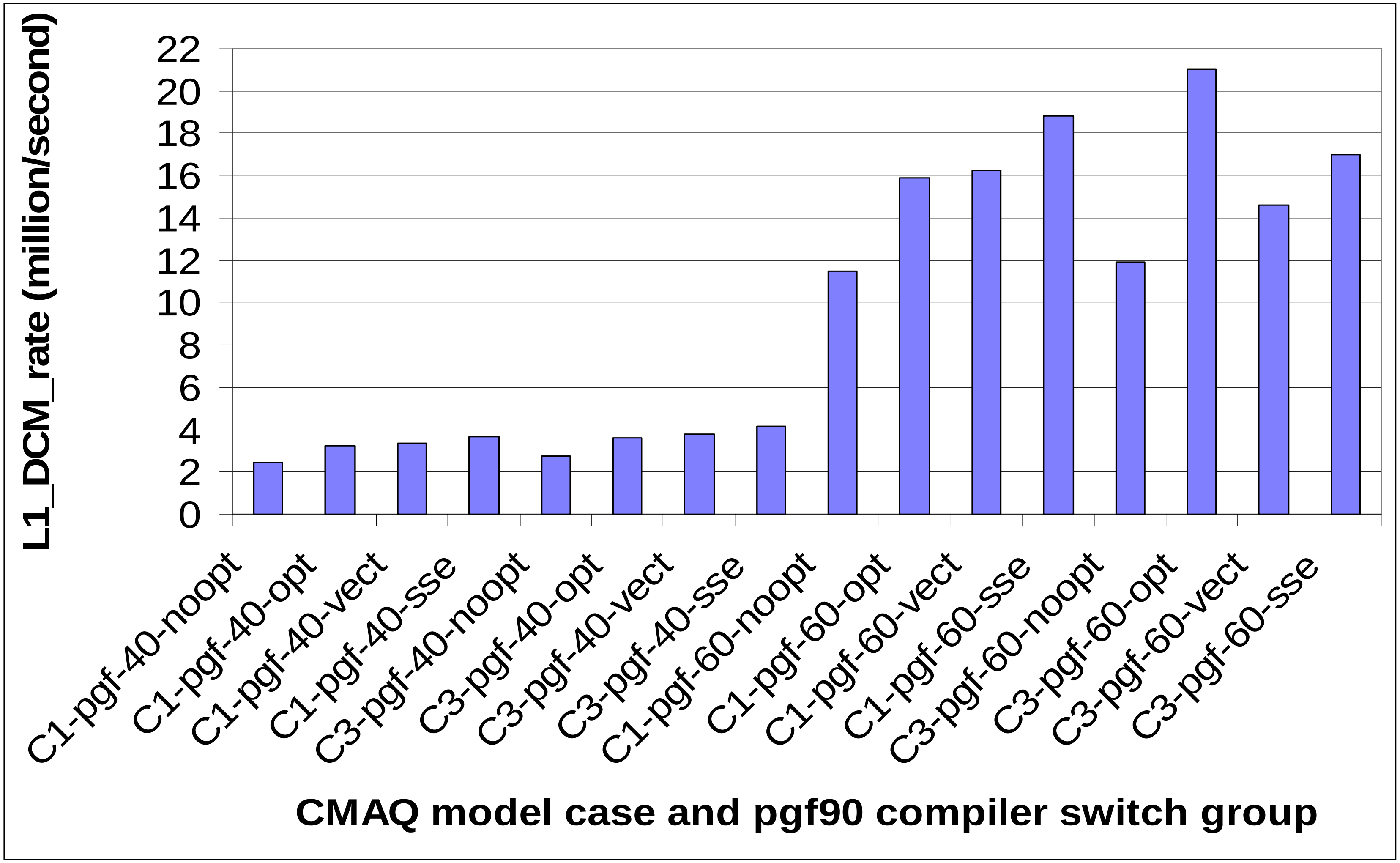
http://www.hiperism.com

6.0 CMAQ Performance Results (data TLB miss rate)

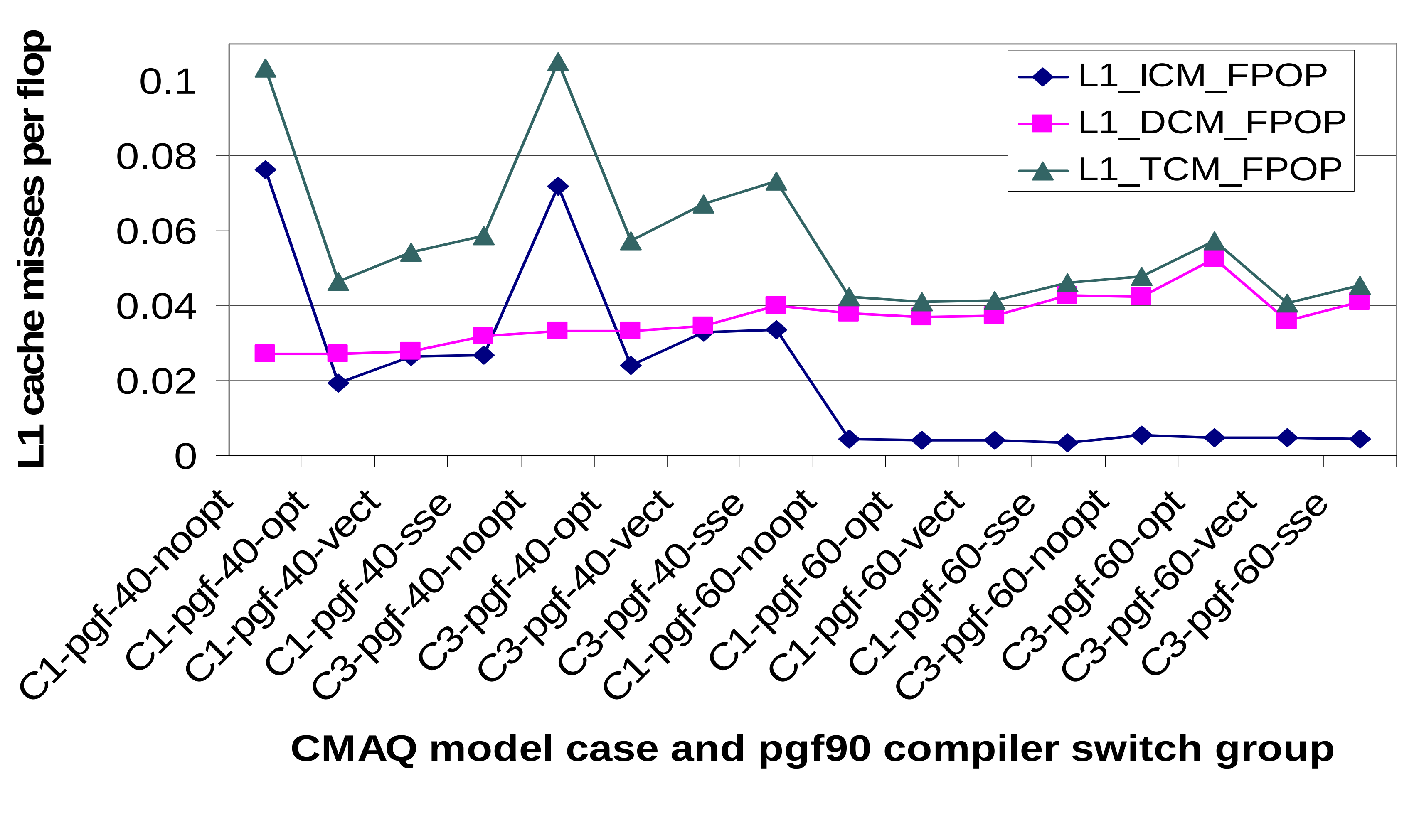


http://www.hiperism.com

6.0 CMAQ Performance Results (L1 data cache miss rate)



6.0 CMAQ Performance Results (L1 cache misses per flop)



http://www.hiperism.com

6. CMAQ Performance Results (Summary)

Ratio of times: for P4 vs P3 was ~ 3x (i.e. clock rate ratio)

- Mflops – 400+ range but:
  - 7% lower Mflops for Case 3 (8km) vs Case1 (32km)
  - Vector SIMD (SSE) instructions give no improvement
- Memory footprint
  - Memory intensive:
    - ~ 4 memory ops per fp operation
    - ~ 1,800 million total memory ops per second !
  - Critical bottle-necks in memory access:
    - Case 3 vs Case 1 has 20% more data TBL miss rates (correlates with 7% Mflops reduction)
    - P4 vs P3 L1 cache misses decline by 10x for instructions, but increase moderately for data misses

7. CMAQ Execution Profile (No. of calls and percent of time)

Function	*No. of calls*	*%*	*No. of calls per second*
hrsolver	2387952	16	
cksummer	2473	15	
hppm	117832	14	
vppm	27461448	9	66,816
ludcmp	2016360	6	4,906
hrcalcks	2387952	4	5,810
calcact	9362289	2	22,779
kmtab	59699539	2	145,254
TOTAL %		***68***	

Summary of Profile Results

- Top two functions
  - use 31% of total time, but
  - source code shows vector instruction inhibition
- Large number of function calls = large overhead for
  - many functions (others not shown) having
    - little computation per call
    - large number of calls

8. Conclusions

- CMAQ shows negligible gain from vector instructions
- CMAQ has ~4 memory operations per floating point operation and large memory access rates. Thus it is memory bandwidth limited on commodity hardware
- CMAQ shows sensitivity to data TLB cache and L1 cache misses and these are serious performance bottlenecks.
- ***CMAQ:*** has problems on commodity hardware with little benefit from the full performance potential.
  - ❑ ***Consequences for CMAQ: some re-design of code to reach potential performance limits is advisable.***