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

HiPERiSM

Models-3 User's Workshop
October 18-20, 2004
Chapel Hill, NC

PERFORMANCE OF AQM APPLICATIONS ON COMMODITY LINUX PLATFORMS

George Delic, Ph.D.

HiPERiSM

Overview

1. Introduction
2. Choice of Hardware, Operating System, & Compilers
3. Choice of Benchmarks
4. Results of Benchmarks
5. Conclusions
6. Outlook

1.0 Introduction

- Motivation
  - AQM's are migrating to COTS hardware
  - Linux is preferred
  - Rich choice of compilers and tools is available
  - Need to learn about portability issues
- What is known about compilers for COTS?
  - Need a requirements analysis of differences in
    - ✓ Performance
    - ✓ Numerical accuracy & stability
    - ✓ Portability issues

2.0 Choice of Hardware, Operating System, and Compilers

- Hardware
  - Intel Pentium 4 Xeon (3 GHz, dual processor) with SSE2 extensions and 1MB L3 cache
  - Linux 2.4.20 kernel
- Fortran compilers for IA-32 Linux
  - Absoft 8.0 (9.0 current release)
  - Intel 8.0 (8.1 released 9/13/04)
  - Lahey 6.2
  - Portland CDK 5.1 (5.2 current release)

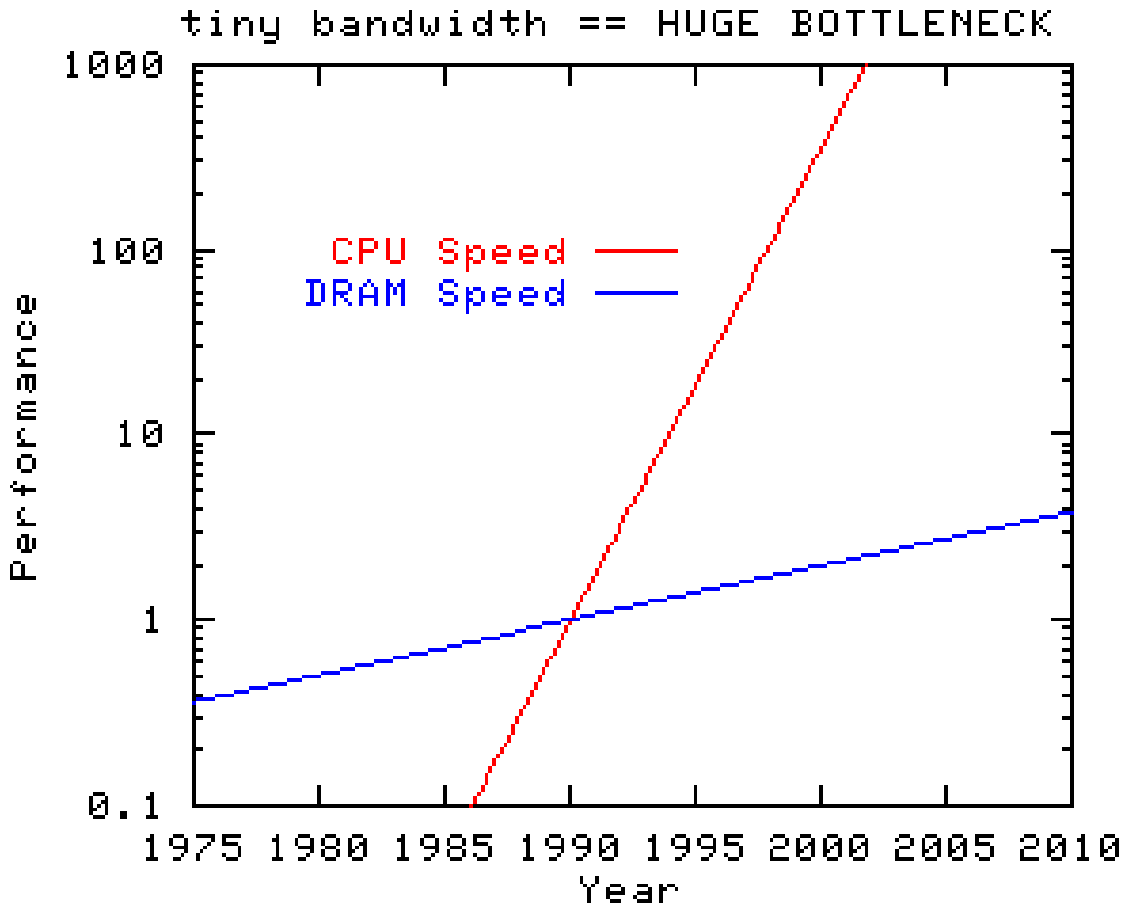
3.0 Choice of Benchmarks

3.1 STREAM memory benchmark

- Developed by John D. McCalpin (VA Tech)
- Available from http://www.streambench.org
- Four kernels to measure memory bandwidth
- Useful on commodity hardware where multiple CPUs share the system bus
- Serial (used here) or OpenMP versions (see HiPERiSM's URL)

The memory bandwidth problem

(STREAM logo used with permission)



 http://www.hiperism.com

HiPERiSM

Compute Kernels of the STREAM benchmark

No	Name	Kernel	Bytes/ iterate	Flops/ iterate	Mops/ iterate
1	COPY	a(i)=b(i)	16	0	2
2	SCALE	a(i)=q*b(i)	16	1	2
3	ADD	a(i)=b(i) + c(i)	24	1	3
4	TRIAD	a(i)=b(i) + q*c(i)	24	2	3

Best results of ten trials with an iteration range of 1 to 20 x 10^6

3.0 Choice of Benchmarks (cont.)

3.2 Princeton Ocean Model (POM)

- Example of "real-world" code that is numerically unstable with sp arithmetic!
- 500+ vectorizable loops to exercise compilers
- 9 procedures account for 85% of CPU time
- 2-Day simulation for three (i,j,k) grids:
  - Grid 1: 100 x 40 x 15 Scaling = 1
  - Grid 2: 128 x 120 x 16 Scaling = 4.4
  - Grid 3: 256 x 256 x 16 Scaling = 17.5

3.0 Choice of Benchmarks (cont.)

3.3 MM5 Community Model v5.3

- History of development on vector machines
- Performance results for many platforms
- Used Storm-of-the-Century (SOC) benchmark
- Compared Intel and Portland compilers

HiPERiSM

3.0 Choice of Benchmarks (cont.)

3.4 CAMx

- Developed by ENVIRON
- Available from http://www.camx.com
- For benchmark used 8/22-8/31, 2000, Scenario for Houston Greater Metro area with TCEQ data
- Used the UT/UNC version of CAMx

Grateful acknowledgements to Dr. Harvey Jeffries and his Ph.D. student, Byeong-Uk Kim, for sharing the code and for help on how to use it.

4.0 Results of Benchmarks

4.1 STREAM results

- Tested with four compilers
- Without and with compiler optimization switches

Note:

STREAM kernels are designed to "fool" optimizing compilers so differences with and without optimization should be small.

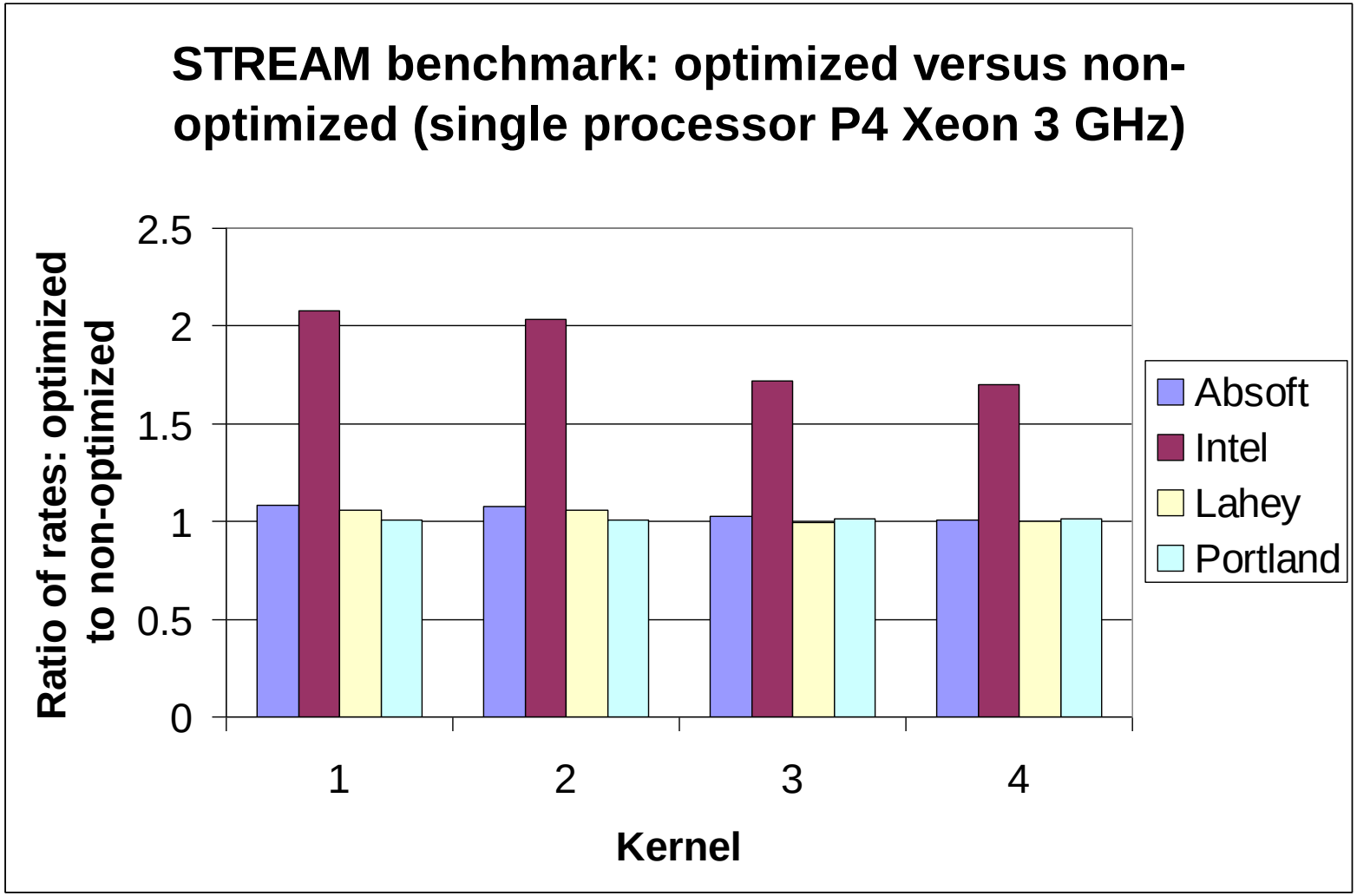
Memory bandwidth for STREAM (MB/second with no optimization)

Name	Absoft	Intel	Lahey	Portland
COPY	1252.3	1289.8	1077.4	1300.8
SCALE	1252.3	1314.2	1138.8	1316.9
ADD	1616.8	1651.8	1230.8	1655.2
TRIAD	1649.4	1650.1	1230.8	1666.7

Memory bandwidth for STREAM (MB/second with optimization)

Name	Absoft	Intel	Lahey	Portland
COPY	1356.6	2677.8	1138.8	1322.3
SCALE	1352.0	2675.6	1207.6	1327.8
ADD	1660.8	2843.6	1227.6	1678.3
TRIAD	1662.7	2802.1	1230.8	1684.21

Memory bandwidth for STREAM: ratio of rates



Summary of STREAM results

- Different compilers produce code with different memory performance
- Performance enhancement with optimization enabled is most noticeable for kernels 1 & 2 with all compilers
- Enabling optimization for the Intel compiler boosts memory bandwidth by factors
  - x 2 for kernels 1 and 2
  - x 1.7 for kernels 3 and 4

Note: these results are specific to serial code – for OpenMP results see the HiPERiSM URL

4.0 Results of Benchmarks (cont.)

4.2 POM results

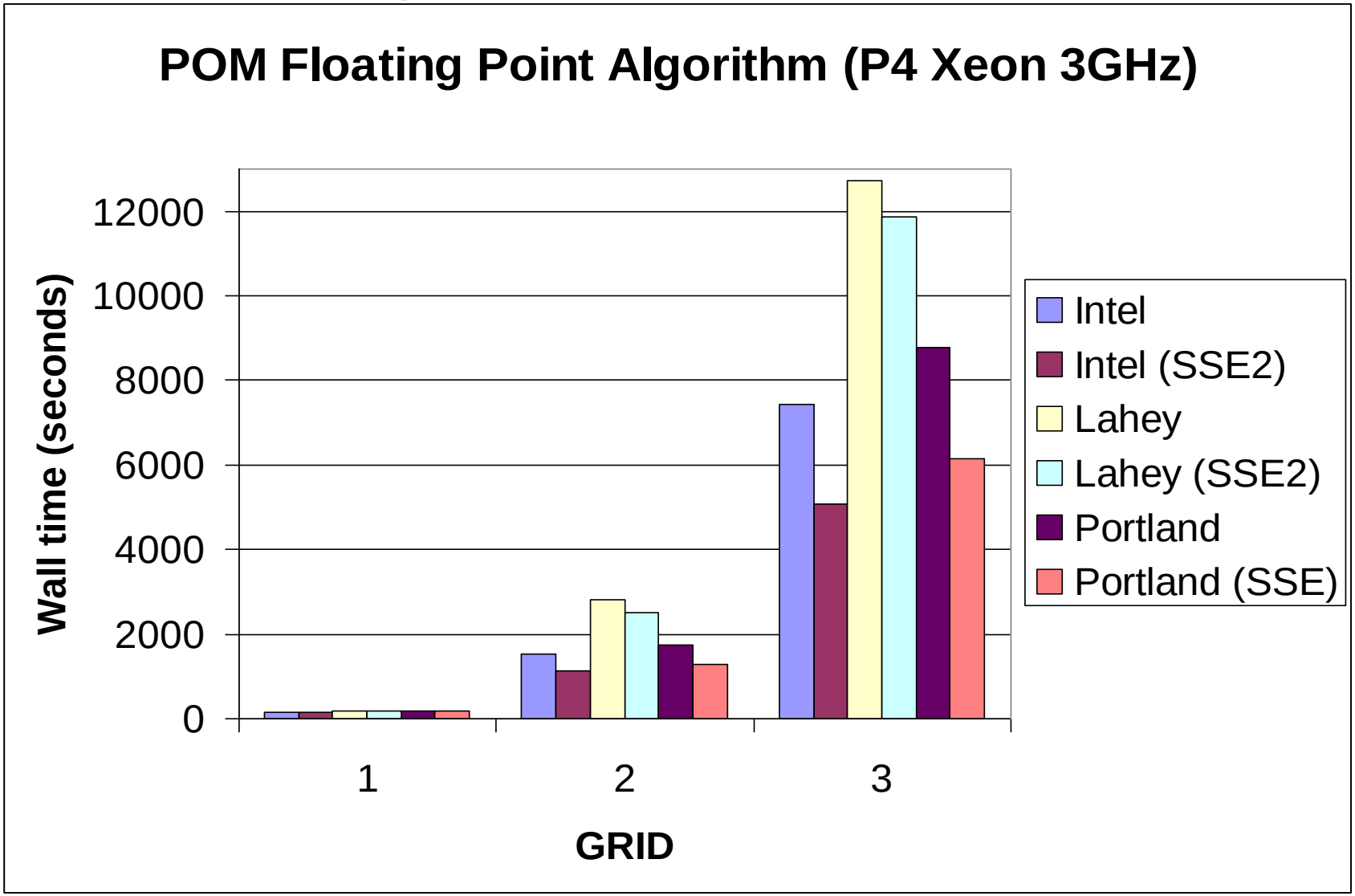
- Tested four compilers without SSE2
- Tested three compilers with SSE2
- Looked at differences with problem size scaling:
  - Grid 1: 100 x 40 x 15 Scaling = 1
  - Grid 2: 128 x 120 x 16 Scaling = 4.4
  - Grid 3: 256 x 256 x 16 Scaling = 17.5

Comparing Execution Times: POM without SSE (seconds)

Grid	Absoft	Intel	Lahey	Portland
1	167.7	156.1	189.4	190.1
2	1925.9	1518.9	2809.7	1756.7
3	8685.2	7432.3	12731.8	8764.8

Note:
The Lahey compiler required the "—long" switch for large integers with larger grids and this reduced performance.

Comparing Execution Times: POM



 http://www.hiperism.com

HiPERiSM

Comparing Execution Times: POM with SSE2 (% gain vs no SSE2)

Grid	Intel	Lahey	Portland
1	1.2	-0.3	11.2
2	25.5	10.3	27.6
3	31.5	6.7	29.8

Summary of POM results

- Without SSE2
  - Grid 2 and Grid 3 show large variability in performance due to Lahey results
  - Grid 1-3 Intel compiler is best (memory boost?)
- With SSE2
  - Performance gain increases with problem size for both Intel and Portland compilers:
    - ~ 26% for Grid 2
    - ~ 30% for Grid 3
  - Intel shows smallest wall clock time (vs. Portland)
    - x 1.1 for Grid 2
    - x 1.2 for Grid 3

4.0 Results of Benchmarks (cont.)

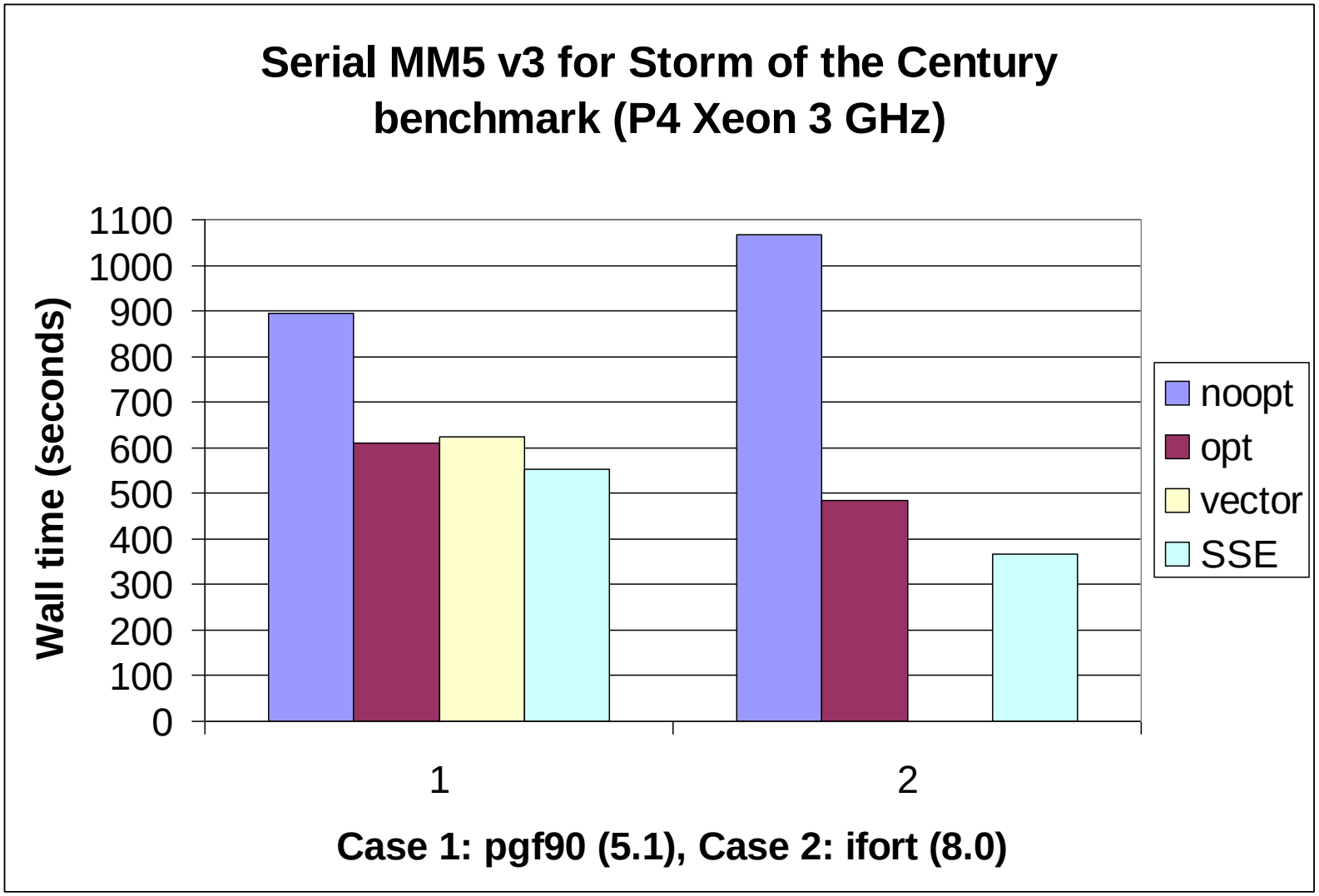
4.3 MM5 Results

- Used Storm-of-the-Century (SOC) benchmark
- Compared Intel and Portland compiler with three switch groups: noopt, opt, SSE

Note:

There is no exact equivalence for the vect group of switches because the Intel compiler does not distinguish vector and SSE instructions.

Comparing Execution Times: MM5



Summary of MM5 results

- Intel performance gain vs noopt
  - Opt switch group yields 55%
  - SSE switch group yields 66%
- Portland performance gain vs noopt
  - opt switch group yields 32%
  - SSE switch group yields 38%
- Speed up of Intel vs Portland
  - for opt switch group x 1.26
  - for SSE switch group x 1.51
- Overall Intel delivers smallest wall clock time

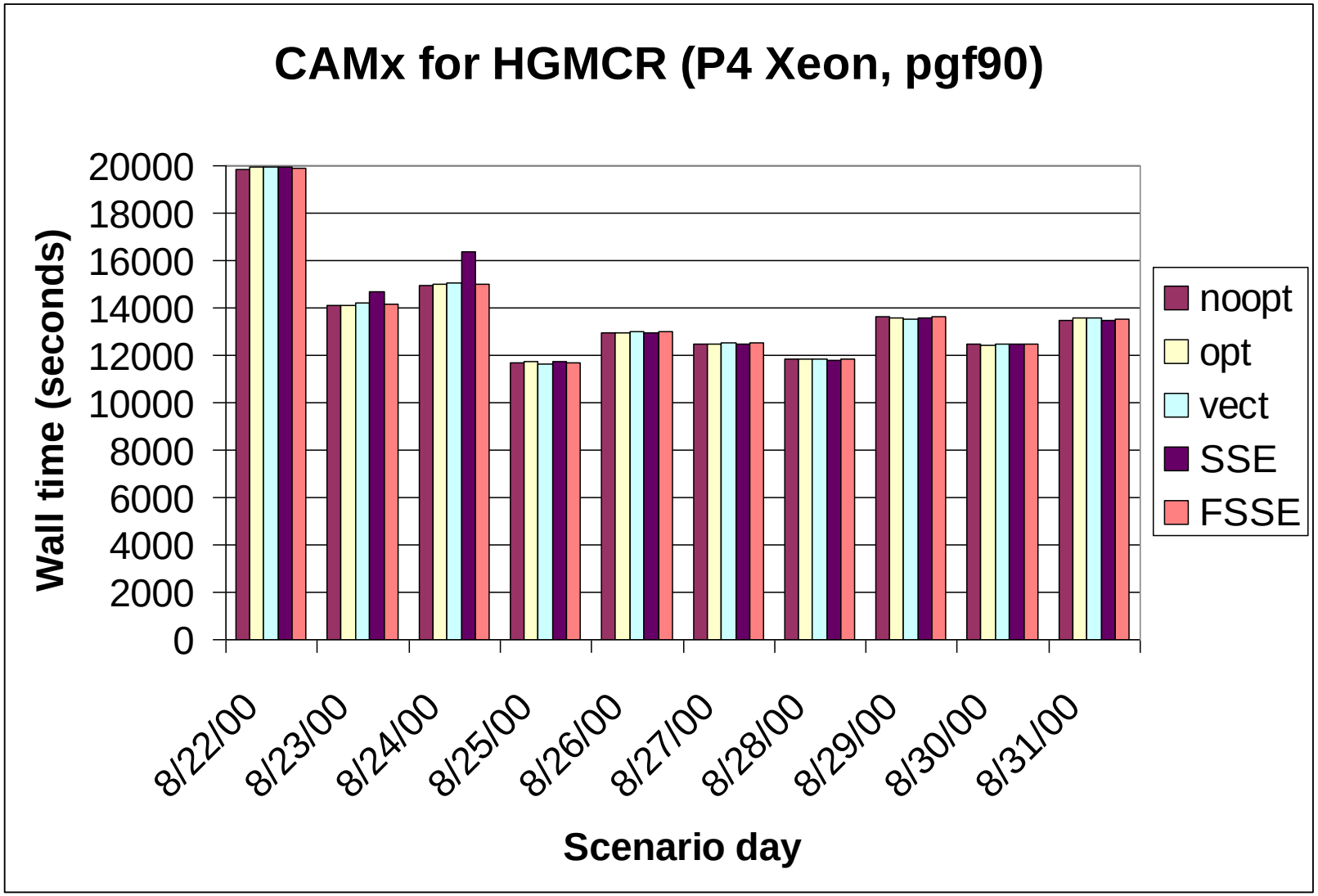
4.0 Results of Benchmarks (cont.)

4.4 CAMx results

- Tested Portland pgf90 compiler with five switch groups that progressively enable higher level optimizations:
  - noopt (baseline: no optimization)
  - opt (scalar optimizations)
  - vect (vector optimizations)
  - SSE (SSE for vector instructions)
  - FSSE (SSE for scalar and vector instructions)

Note:
Portland compiler technical support made some suggestions.

Comparing Execution Time: CAMx



Summary of CAMx results

- NO REDUCTION IN WALL CLOCK TIME FROM ANY COMPILER OPTIMIZATIONS !!!!!
- CAMx receives no benefit from this generation of hardware and compiler technology which has been demonstrated to produce such benefits in other cases.
- Possible reasons:
  - This is scalar code and vector potential is inhibited
  - CAMx is I/O bound and writes ONE word at a time in implicit nested loops !!!!!!!!!!!!
  - CAMx is sensitive to memory and I/O bandwidth.

5.0 Conclusions

- ***STREAM*** shows that memory bandwidth is the principal differentiator for COTS (not clock speed).
- ***POM and MM5*** showed that legacy vector codes receive performance boosts from optimizations and SSE on COTS hardware with current compilers and performance gains increase with larger problem sizes
- ***CAMx:*** has serious problems on COTS and does not benefit from the performance potential available now.
  - ***Consequences for CAMx: re-design of code to reach potential performance limits is advisable.***

6.0 Outlook

- ***Hardware:*** COTS is delivering good performance on legacy vector code. The outlook is good for such code since future HPC CPU's will have longer pipelines and larger cache.
- ***Linux:*** Operating System is sufficiently reliable.
- ***Programming Environment:*** rich in compiler and tools technology for code developers.
- ***Consequences for AQM:*** the outlook for hardware, Linux, and programming environment requires ***careful on-going re-design of code to reach potential performance limits of future COTS technology.***

HiPERiSM's URL

http://www.hiperism.com

Technical Reports pages for details of the compiler switches

