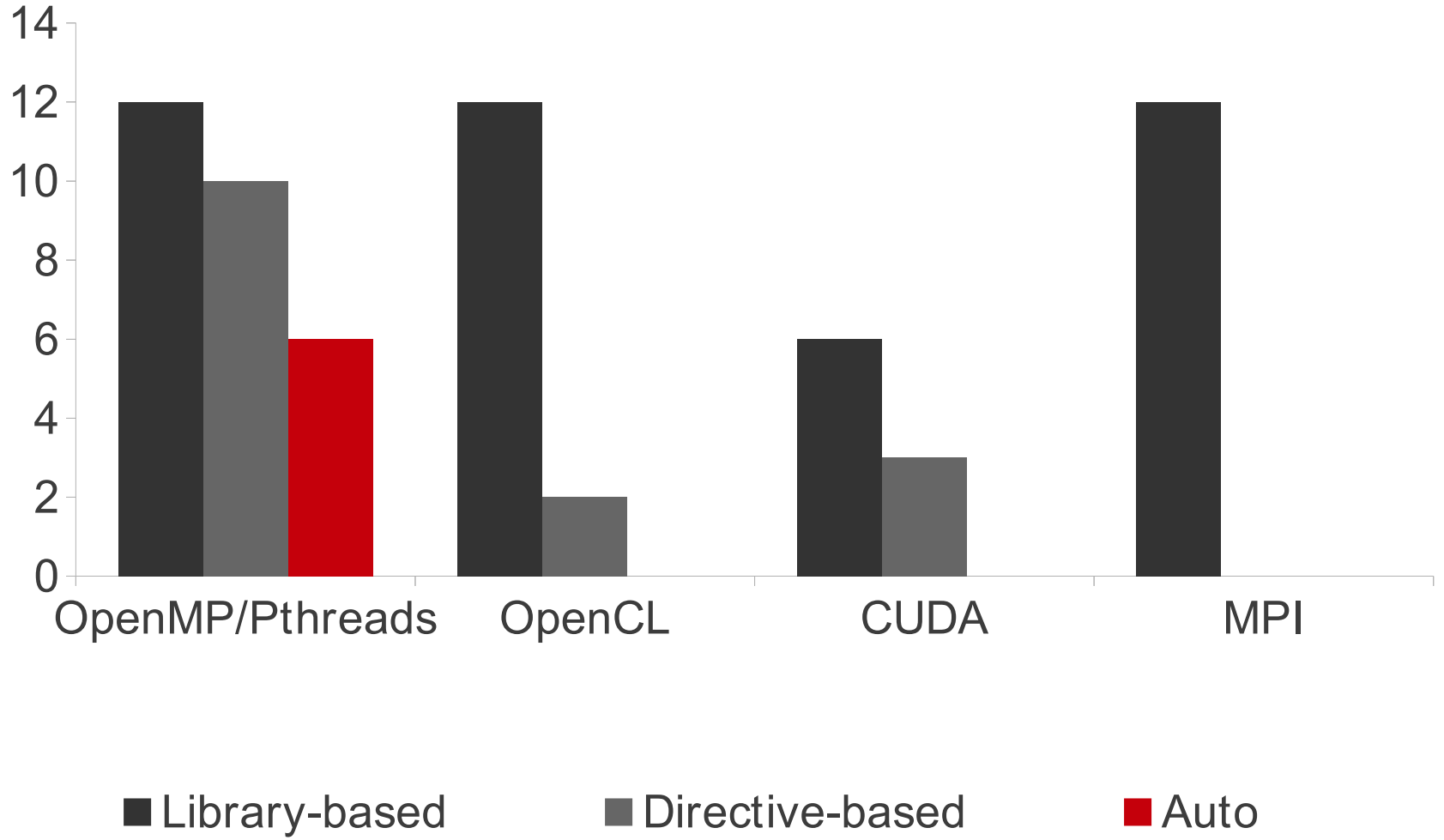
Practical Efficiency of Optimizing Compilers in Parallel Scientific Applications

Bogdanov A.V., Gankevich I.G.
Saint-Petersburg State University
bogdanov@csa.ru, gig.spb@gmail.com

Outline

- Compilers overview
- Micro benchmarks results
- Parallel benchmarks results
- Conclusion

Parallel standards support



Compilers being analyzed: GCC, Intel, PGI, Clang, Lahey, NAG, CAPS HMPP, Pathscale, Absoft, Silverfrost FTN95, MSVC, IBM XL

Compilers being benchmarked

- GCC, Intel, PGI
  - All support OpenMP
  - All support autoparallelization on CPU
  - PGI supports directive based parallelization on GPU

Parallelization benchmark setup

- Source code
  - Reference BLAS implementation (www.netlib.org) compiled with automatic parallelization options
  - Hand-tuned Goto BLAS implementation compiled with library-chosen optimization options
- Target platform
  - HP SL390s G7
  - 2x Intel X5650 2.67 Ghz (12 cores)
  - 96 Gb RAM
  - 3x NVIDIA Tesla M2050

Parallelization benchmark setup

- Routines being analyzed
  - SGEMM, SSYMM, SSYR2K, SSYRK, STRMM, STRSM
- Variations
  - Different sizes (512, 1024, 2048, 4096, 8192)
  - Transposed/nontransposed matrices
  - Upper/lower triangular matrices
  - Left/right sided
- A total of 128 test cases were analysed for each compiler and different amount of threads

Average performance

	Hand-tuned Goto BLAS			Autoparallelized BLAS (CPU)			Directive-parallelized BLAS (GPU)
	Intel	PGI	GCC	Intel	PGI	GCC	PGI
SGEMM	111974	**130197**	57816	**16173**	7333	1824	1939
SSYMM	108611	**130466**	60525	9089	**12123**	2390	1632
SSYR2K	103813	**112828**	60230	**18157**	13728	3136	1917
SSYRK	95721	**106000**	43239	**14562**	10027	1811	1218
STRMM	99073	**115354**	43204	**14423**	7148	1990	2102
STRSM	97371	**115009**	41857	**11381**	6295	1745	1998

Performance is reported in Mflops as the average of all routine arguments variations

SGEMM source code

SGEMM(A^T, B)	SGEMM(A^T, B^T)

DO 120 J = 1,N
  DO 110 I = 1,M
  TEMP = ZERO
  DO 100 L = 1,K
  TEMP = TEMP +
A(L,I)*B(L,J)
  CONTINUE
  IF (BETA.EQ.ZERO) THEN
  C(I,J) = ALPHA*TEMP
  ELSE
  C(I,J) = ALPHA*TEMP +
BETA*C(I,J)
  END IF
  CONTINUE
CONTINUE
	DO 200 J = 1,N
	  DO 190 I = 1,M
	  TEMP = ZERO
	  DO 180 L = 1,K
	  TEMP = TEMP +
	A(L,I)*B(J,L)
	  CONTINUE
	  IF (BETA.EQ.ZERO) THEN
	  C(I,J) = ALPHA*TEMP
	  ELSE
	  C(I,J) = ALPHA*TEMP +
	BETA*C(I,J)
	  END IF
	  CONTINUE
	CONTINUE

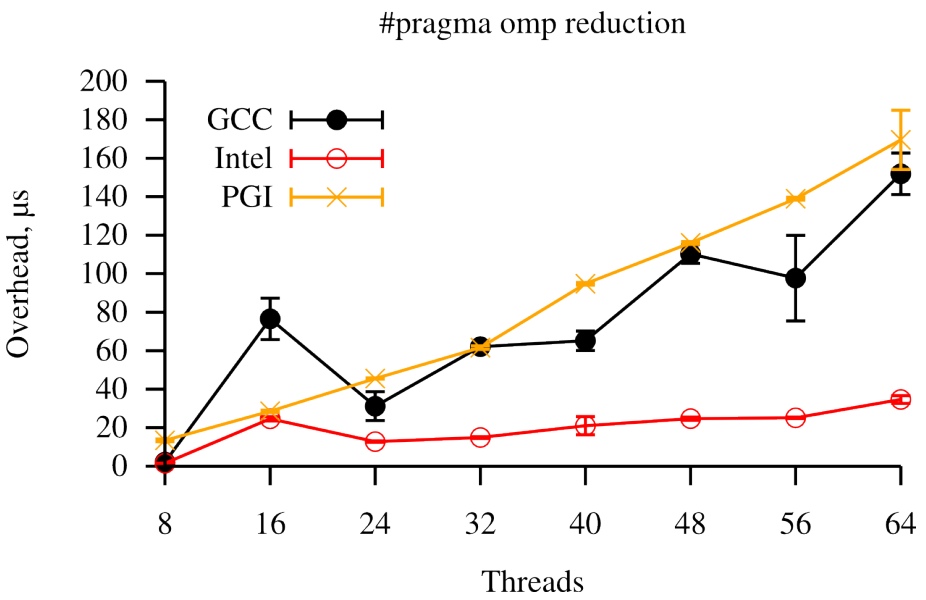
Memory performance

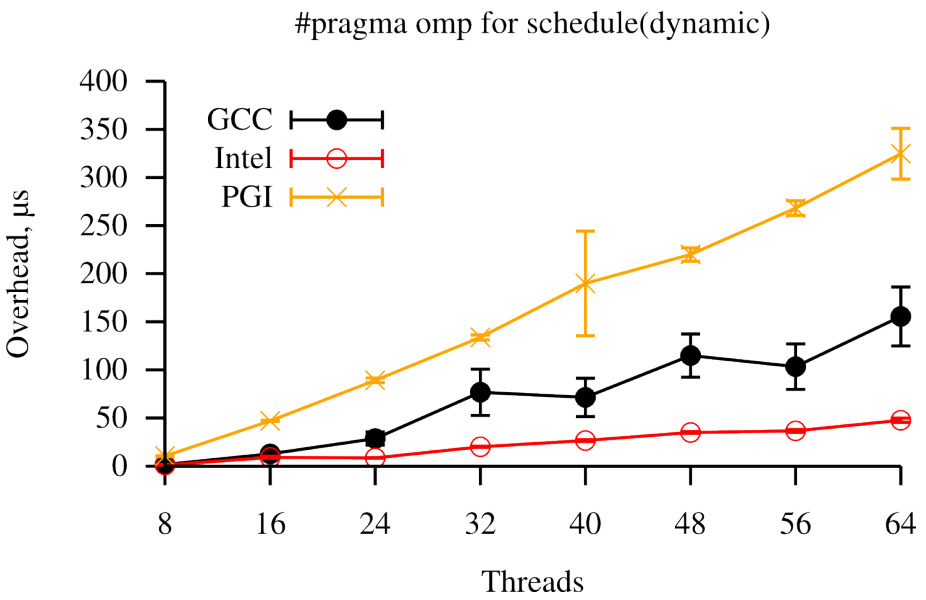
	Reads $\times 10^9$	Writes $\times 10^9$	Instructions $\times 10^9$	Misses $\times 10^9$
SGEMM(A^T, B)	137.4	0.016	412	0.008
SGEMM(A^T, B^T)			549	0.104

OpenMP benchmark setup

- Source code
  - EPCC benchmarks compiled with maximum optimization level
- Target platform
  - HP Proliant DL980
  - 8x Intel X7560 2.2 Ghz (64 cores)
  - 512 Gb RAM

OpenMP directives overhead





Conclusions

- Autoparallelization on CPU is suitable for prototyping
- Directive-based parallelization on GPU is immature and requires hand tuning

Thank you