Tema 2 ASC

336 CC

Bînă Marius-Andrei

Detalii implementare:

1. Soluție Neoptimizată
Pentru soluția neoptimizată am folosit cea mai basic abordare, 3 for-uri fără salvare de indecși pentru acces mai eficient al elementelor, salvare de pointeri sau alte metode de optimizare, de asemenea se compilează cu flag -O0 (fără optimizări realizate de compilator). Cum se va observa în continuare, această abordare are, evident, cea mai slabă performanță.

2. Soluție Optimizată
Pentru soluția optimizată am pornit de la cea neoptimizată și am îmbunătățit-o pas cu pas. Primul pas a fost folosirea unor variabile marcate ca registre pentru salvarea sumelor, folosind aceste variabile, memoria nu mai este accesata pentru matrice la fiecare for interior (k), ci se aduna de fiecare data în aceste sume, iar după terminarea for-ului se actualizează în memorie astfel este un singur acces pe o bucla a forului j, fața de N accesări. Următorul pas a fost salvarea indecșilor folosiți pentru accesul memoriei, astfel, unde nu este nevoie să fie calculati la fiecare iteratie, sunt calculati înainte și folosiți direct, exemplu face index-ul de acces (i, j), dar și i_N și j_N care ajuta la realizarea inmultirii de mai puține ori. Ultimul pas și cel mai important a fost folosirea unor pointeri pentru eficientizare accesului la memorie, astfel am obținut o performanța sporita, pointerii sunt în seturi de câte doi (parte reala și parte imaginara) și în forul interior sunt deplasati cu doua poziții (următoarea poziție după partea reala fiind partea imaginara a aceluiasi numar).

3. Soluție Optimizată la Compliare
Compilarea solutiei neoptimizate cu optiunea de optimizare -O3 a adus performanțe mai bune decât soluția optimizata din codul C prezentata mai sus datorită folosirii vectorizarilor efectuate și a altor opțiuni de optimizare din interiorul acesteia.

4. BLAS
Cele mai bune performanțe au fost obtinute folosind functia blas_zsyrk din biblioteca blas.h, datorită unei implementari în Fortran. Pentru a folosi aceasta funcție a fost nevoie de linkare cu BLAS Atlas prin completarea caii de biblioteca în Makefile-uri cu /usr/lib64/atlas/libsatlas.so.3.10.

Statistici:

GCC	GCC NEOPT	GCC OPT_M	GCC OPT_F	GCC BLAS
	10,475473	4,383218	1,189829	0,494076
	18,532495	7,685669	2,281562	0,844435
	29,870293	12,127424	3,765911	1,341902
	44,127964	17,932116	5,021067	2,018558
	62,427769	24,501856	8,740804	2,919575
AVERAGE	33,0867988	13,326057	4,1998346	1,5237092
IMPROVEMENT	-	59,72%	87,31%	95,39%

	GCC	ICC	GCC > ICC
NEOPT	33,0867988	34,486695	4,06%
OPT_M	13,326057	14,3915656	7,40%
OPT_F	4,1998346	5,5102456	23,78%
BLAS	1,5237092	1,5460296	1,44%
AVERAGE	13,0340998	13,98363395	6,79%

ICC	ICC NEOPT	ICC OPT_M	ICC OPT_F	ICC BLAS
	11,14035	4,738421	1,722681	0,516744
	19,496628	7,974349	2,874223	0,882538
	30,501209	12,30085	4,605141	1,386519
	45,480934	19,348207	7,179817	2,061663
	65,814354	27,596001	11,169366	2,882684
AVERAGE	34,486695	14,3915656	5,5102456	1,5460296
IMPROVEMENT	-	58,27%	84,02%	95,52%

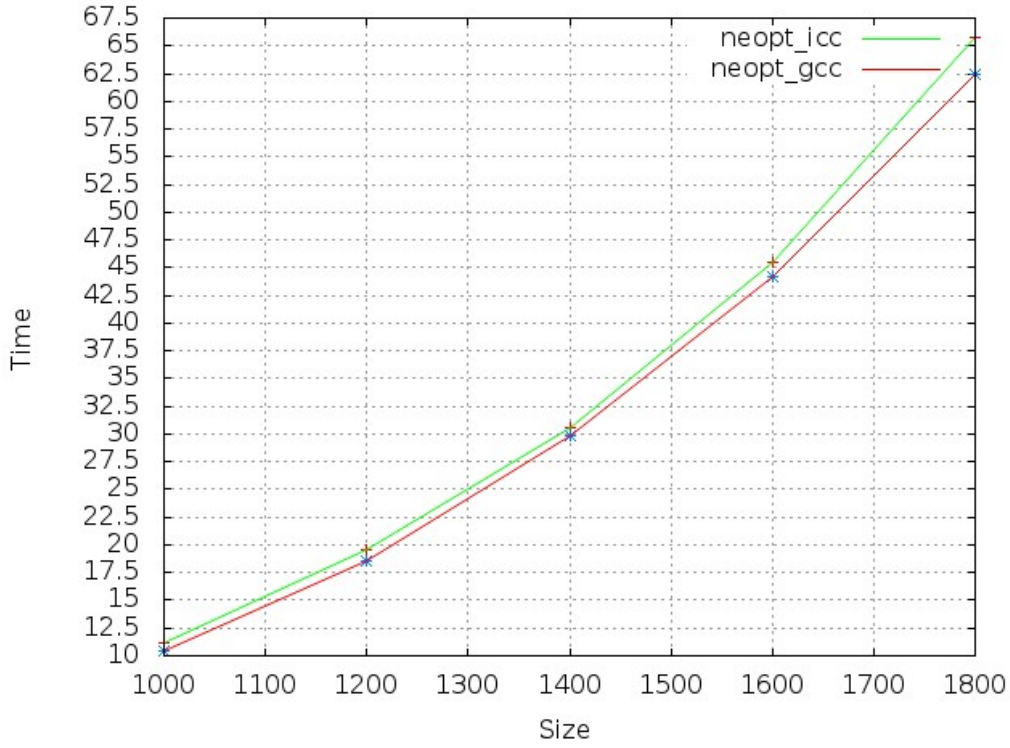
In imaginea de mai sus se pot observa timpii de executie pe teste pentru fiecare metoda si pentru ambele compilatoare. In linia **AVERAGE** se gaseste media de timp pentru fiecare metoda pe toate cele 5 teste din input. Folosind aceasta medie, am calculat procentul cu care s-a imbunatatit la fiecare metoda fata de cea mai slaba metoda, cea neoptimizata. Acest lucru a fost facut pentru ambele compilatoare, dupa cum se poate observa.

ICC vs GCC

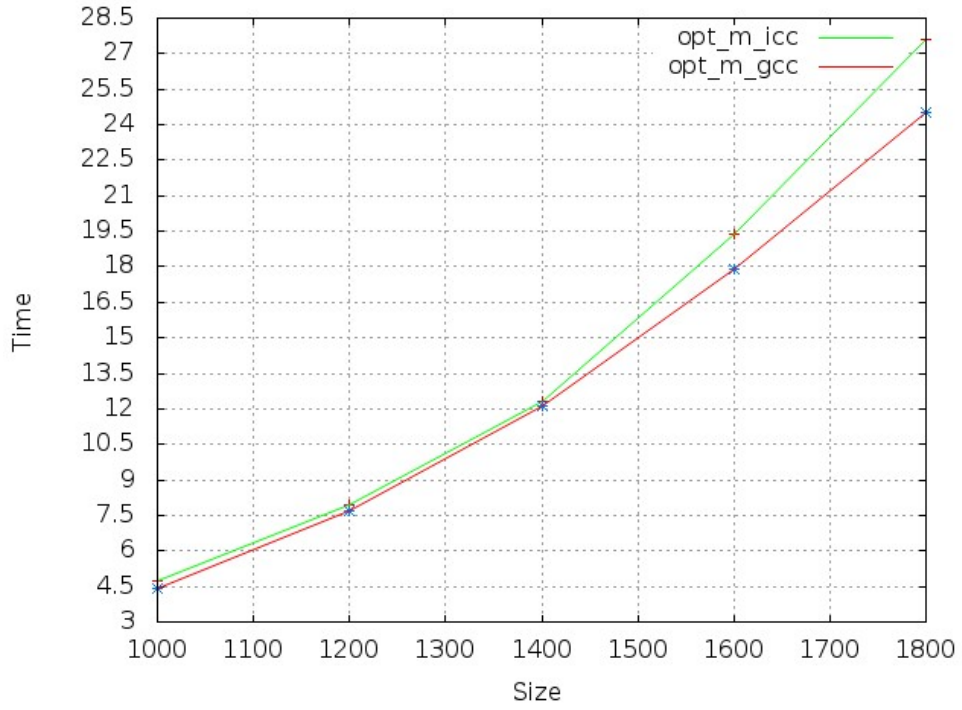
In partea dreapta a imaginii de mai sus se observa o comparatie intre mediile obtinute pe cele 5 teste in cele 4 metode. In afara metodei in care am compilat solutia neoptimizata cu flag de optimizare pentru compilator, diferenta nu este majora, avand un maxim de 7.40% in favoarea GCC-ului. In cazul metodei OPT_F diferenta este destul de mare, 23.78%, iar in timp asta se traduce in mai mult de o secunda. Acest lucru se datoreaza faptului ca flagul -O3 pentru GCC contine intern mai multe optiuni de optimizare decat in cazul lui ICC.

Aceste analize si statistici se pot observa in urmatoarele grafice comparative:

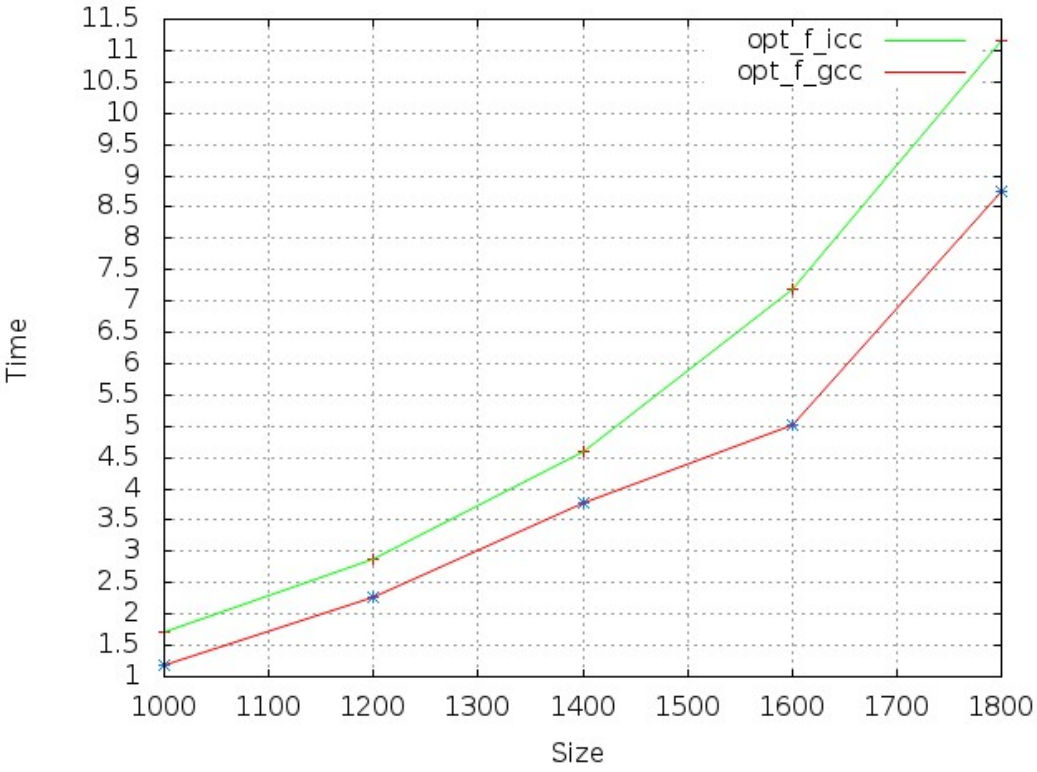
1. ICC vs GCC - NEOPT



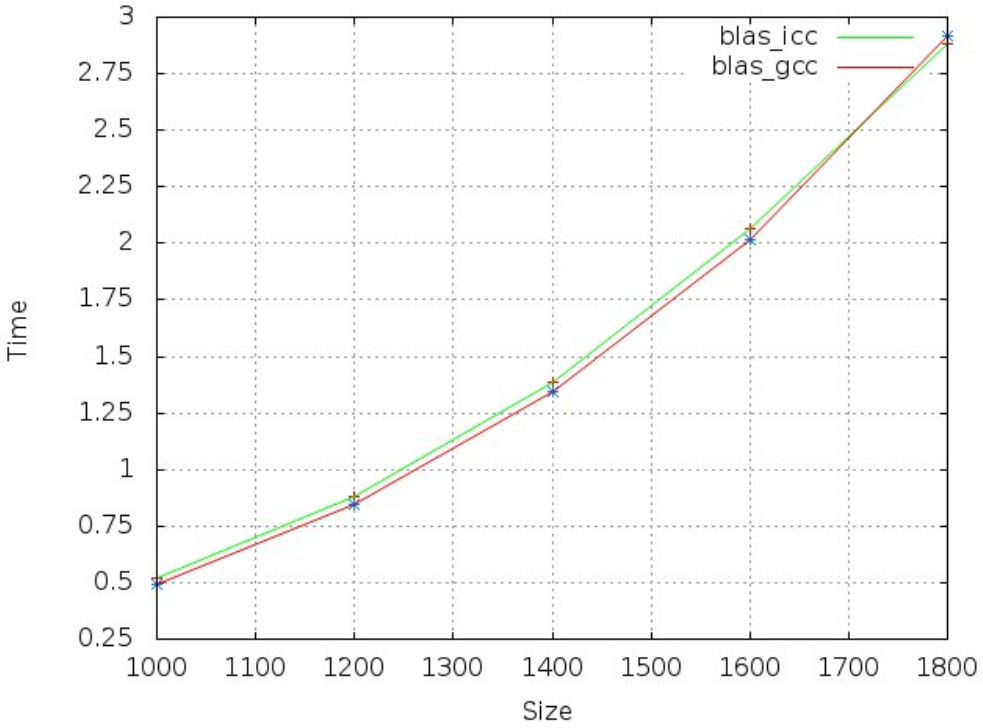
2. ICC vs GCC – OPT_M



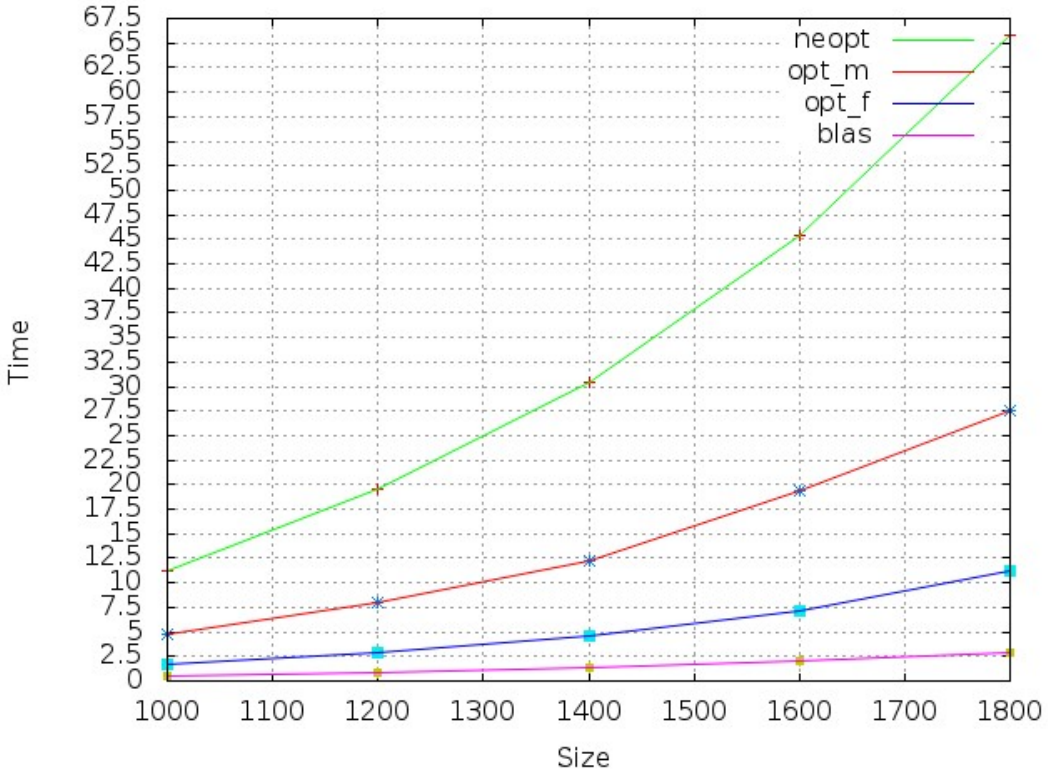
3. GCC vs ICC – OPT_F



4. GCC vs ICC – BLAS



5. ICC – Cele 4 metode



6. GCC – Cele 4 metode