Tema 2 ASC

April 1, 2018

1 Enunt

Se da urmatoarea operatie cu matrice:

$$C = A \times A^t$$

unde:

- A este o matrice de numere complexe (a + b $\times$ i, unde a si b sunt de tip double) de dimensiune NxN
- A^t este transpusa lui A
- $\times$ este operatia de inmultire

Se doreste implementarea operatiei de mai sus in C/C++ in 4 moduri:

- **blas** - o varianta care foloseste functia **zsyrk** din BLAS Atlas [1] [2]
- **neopt** - o varianta "de mana" fara imbunatatiri
- **opt_m** - o varianta imbunatatita a versiunii de mai sus. Imbunatatirea are in vedere exclusiv modificarea codului pentru a obtine performante mai bune.
- **opt_f** - o alta varianta imbunatita obtinuta prin compilarea codului de la varianta **neopt** cu alte flag-uri de compilare ce aduc un bonus de performanta

Avand in vedere ca rezultatul operatiei cerute este o matrice simetrica se va completa doar partea de deasupra diagonalei principale (C este matrice superior triunghiulara).

2 Rulare si testare

Pentru testarea temei va este oferit un schelet de cod pe care trebuie sa-l completati cu implementarile pentru cele 4 variante mentionate mai sus. Scheletul de cod este structurat astfel:

[1] `http://www.netlib.org/blas/`

[2] `http://www.netlib.org/lapack/explore-html/de/d54/zsyrk_8f_source.html`

- **main.c** contine functia main, precum si alte functii folosite pentru citirea fisierului cu descrierea testelor, scrierea matricei rezultat intr-un fisier, generarea datelor de intrare si rularea unui test. Acest fisier va fi suprascris in timpul corectarii si nu trebuie modificat.
- **utils.h** fisier header. Acest fisier va fi suprascris in timpul corectarii si nu trebuie modificat.
- **solver_blas.c** - in acest fisier trebuie sa adaugati implementarea variantei **blas**
- **solver_neopt.c** - in acest fisier trebuie sa adaugati implementarea variantei **neopt**
- **solver_opt.c** - in acest fisier trebuie sa adaugati implementarea variantei **opt_m**
- **Makefile** - Makefile folosit la compilarea cu gcc
- **Makefile.icc** - Makefile folosit la compilarea cu icc

Puteti aduce orice modificare scheletului de cod exceptand cele 2 fisiere mentionate mai sus.
In urma rularii comenzii **make** cu oricare din cele 2 Makefile-uri vor rezulta 4 fisiere binare, **tema2_blas**, **tema2_neopt**, **tema2_opt_m** si **tema2_opt_f** corespunzatoare celor 4 variante care trebuie implementate.
Rularea se va realiza astfel:

```
./tema2_blas input
```

unde **input** este fisierul ce contine descrierea testelor si este structurat astfel: pe prima linie numarul de teste, pe urmatoarele linii descrierea fiecarui test reprezentata de valoarea lui N, seed-ul folosit la generarea datelor si calea catre fisierul de iesire ce contine matricea rezultat.
Rularea se face in acelasi mod si pentru **tema2_neopt**, **tema2_opt_m** si **tema2_opt_f**.
Fisierul de test folosit pentru validarea celor 4 variante este prezent in arhiva cu scheletul de cod. Continutul este urmatorul:

```
5
1000 1234 out1
1200 5678 out2
1400 9012 out3
1600 3456 out4
1800 7890 out5
```

Rularea se va face pe coada **ibm-dp.q**. Compilarea se va face folosind urmatoarele compilatoare:

- gcc 5.4.0 din modulul compilers/gnu-5.4.0
- icc 16.0.2 din modulul utilities/intel_parallel_studio_xe_2016

Exceptand varianta **opt_f** nu se vor folosi flag-uri de compilare pentru optimizari.
Pentru linkarea cu BLAS Atlas se va folosi versiunea single-threaded (libsatlas.so.3.10) din /usr/lib64/atlas atat pentru gcc, cat si pentru icc.
Fisierele output referinta le gasiti aici: **/export/asc/tema2/**

3 Punctaj

Punctajul este impartit astfel:

- 15p pentru implementarea variantei **blas**
- 15p pentru implementarea variantei **neopt**
- 15p pentru implementarea variantei **opt_m**
- 15p pentru implementarea variantei **opt_f** dintre care 7.5p pentru gcc si 7.5p pentru icc
- 10p pentru descrierea implementarii pentru fiecare din cele 4 variante (2.5p pentru fiecare varianta)
- 30p pentru analiza performantei dintre care:
  - 10p pentru analiza comparativa **blas** vs **neopt** vs **opt_m** vs **opt_f** a performantei pentru **gcc (5p)** si **icc (5p)**
  - 10p pentru o analiza comparativa **icc** vs **gcc**
  - 10p pentru realizarea unor grafice relevante pe care sa se bazeze analizele de mai sus
- (Bonus) 20p daca varianta **opt_m** este de cel mult 4 ori mai lenta decat BLAS pe toate cele 5 teste folosind oricare din cele 2 compilatoare

Pentru a fi luata in considerare la punctaj implementarea trebuie sa produca rezultate corecte pe toate cele 5 teste.

4 Precizari si recomandari

Timpul maxim pentru rularea celor 5 teste folosind oricare din cele 4 variante este de 4 minute. Aceasta limita de timp se refera la rularea intregului program, nu doar la partea intensiv computationala.
Pentru a simplifica implementarea puteti presupune ca N este multiplu de 40 si ca este mai mic de 2000.
In compararea rezultatelor se va permite o eroare absoluta de maxim 10^{-3}.
In cazul variantei **opt_m** complexitatea trebuie sa fie aceeasi cu cea din varianta **neopt**.
In cazul variantei **opt_f** se pot folosi atat flaguri de optimizare generice (O1/O2/O3) cat si flag-uri specifice ce nu sunt restrictionate de folosirea unui anumit nivel de optimizare. [3] [4]

Pentru analiza performantei puteti folosi si alte date de intrare decat cele 5 teste din schelet.
Un numar complex este stocat in memorie ca doua double-uri consecutive.
Se recomanda stergerea fisierelor coredump in cazul rularilor care se termina cu eroare pentru a evita problemele cu spatiul de stocare.

[3] `https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html`
[4] `https://software.intel.com/en-us/articles/step-by-step-optimizing-with-intel-c-compiler`