Tema 2

- **Deadline soft:** 04 mai 2022 ~~02 mai~~, ora 23:55. Primiți un bonus de 10% din punctajul obtinut pentru trimiterea temei înainte de 29 aprilie 2022, ora 23:55.
- **Deadline hard:** 09 mai 2022, ora 23:55. Veți primi o depunctare de 10% din punctajul maxim al temei pentru fiecare zi de întârziere (dupa ~~02 mai~~ 04 mai), până la maxim ~~7 zile~~ 5 zile, adică până pe 09 mai 2022, ora 23:55.
- **Responsabili:** Vlad Spoiala [mailto:vladspoiala@gmail.com], Cosmin-Gabriel Samoila [mailto:gabrielcsmo@gmail.com], Emil Slusanschi [mailto:emil.slusanschi@cs.pub.ro], Radu Petru Daia [mailto:radu_petru.daia@stud.acs.upb.ro]

- Dată publicare: 16 aprilie 2022
- Dată actualizare enunț: 02 mai 2022

Enunț

Se dă următoarea operație cu matrice:

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

unde:

- A si B sunt matrice patratice de double de dimensiune N x N
- A **este o matrice superior triunghiulara**
- A^t este transpusa lui A si B^t este transpusa lui B
- $\times$ este operația de înmulțire
- $+$ este operatia de adunare

Se dorește implementarea operației de mai sus in C/C++ în 3 moduri:

- **blas -** o variantă care folosește una sau mai multe functii din BLAS Atlas [http://www.netlib.org/blas/] pentru realizarea operatiilor de inmultire si adunare.
- **neopt -** o variantă "de mână" fără îmbunătățiri.
- **opt_m -** o variantă îmbunătățită a versiunii **neopt**. Îmbunătățirea are în vedere exclusiv modificarea codului pentru a obține performanțe mai bune.

Fiecare din cele 3 implementari de mai sus va tine cont de faptul ca **A este o matrice superior triunghiulara**.

Rulare și testare

Pentru testarea temei vă este oferit un schelet de cod pe care trebuie să-l completați cu implementarile celor 3 variante menționate mai sus. Scheletul de cod este structurat astfel:

- **main.c -** conține funcția main, precum și alte funcții folosite pentru citirea fișierului cu descrierea testelor, scrierea matricei rezultat într-un fișier, generarea datelor de intrare și rularea unui test. Acest fișier va fi suprascris în timpul corectării și nu trebuie modificat.
- **utils.h -** fișier header. Acest fișier va fi suprascris în timpul corectării și nu trebuie modificat.
- **solver_blas.c -** în acest fișier trebuie să adaugați implementarea variantei **blas**.
- **solver_neopt.c -** în acest fișier trebuie să adaugați implementarea variantei **neopt**.

- **solver_opt.c -** în acest fișier trebuie să adaugați implementarea variantei **opt_m**.
- **Makefile** - Makefile folosit la compilarea cu gcc.
- **compare.c** - utilitar ce poate fi folosit pentru a compara doua fisiere rezultat. Acest fișier va fi suprascris în timpul corectării și nu trebuie modificat**.**

Puteți aduce orice modificare scheletului de cod exceptând cele 3 fișiere menționate mai sus.

În urma rulării comenzii **make** vor rezulta 3 fișere binare, **tema2_blas**, **tema2_neopt** si **tema2_opt_m** corespunzătoare celor 3 variante care trebuie implementate.

Rularea se va realiza astfel:

```
./tema2_<mod> input
```

unde:

- mod este unul din modurile **blas**, **neopt**, **opt_m**
- input este fișierul ce contine descrierea testelor.

Fișierul **input** este structurat astfel:

- pe prima linie numărul de teste.
- pe următoarele linii descrierea fiecarui test:
  - valoarea lui N.
  - seed-ul folosit la generarea datelor.
  - calea către fișierul de ieșire ce conține matricea rezultat.

Rularea se va face pe partitia **nehalem**. Compilarea se va face folosind **gcc-8.5.0**. Pentru variantele **blas**, **neopt** si **opt_m** nu vor fi utilizate flag-uri de optimizare pentru compilare (va fi utilizat -O0). Pentru linkarea cu BLAS Atlas se va folosi versiunea single-threaded **libsatlas.so.3.10** disponibila in directorul

```
/usr/lib64/atlas
```

de pe masinile din partitia nehalem

Nodurile din partitia nehalem sunt mai lente decat nodurile din alte partitii. Testele voastre de performanta trebuie realizate pe partitia nehalem deoarece evaluarea temelor se va face pe aceasta partitie.

Fișierele input ce vor fi folosite pentru testare si fisierele output referință le găsiți pe fep in acest folder:

```
 /export/asc/tema2/
```

Fisierul **input** contine 3 teste:

```
3
400 123 out1
800 456 out2
1200 789 out3
```

In cazul fisierului **input** avem 3 teste pentru urmatoarele valori ale lui N: 400, 800, respectiv 1200. Seed-urile folosite la generarea datelor de intrare sunt 123, 456, respectiv 789. Fisierele de output sunt out1, out2, respectiv out3.

Pentru a fi luată în considerare la punctaj, implementarea trebuie să producă rezultate corecte pe toate cele 3 teste din fisierul **input**.

Fisierul **input_valgrind** ce va fi folosit pentru rularile de valgrind contine un singur test:

```
1
400 123 out1
```

Punctaj

Punctajul este impărțit astfel:

- **15p** pentru implementarea variantei **blas** dintre care:
  - 12p daca implementarea obtine rezultate corecte
  - 3p pentru descrierea implementarii in README
- **15p** pentru implementarea variantei **neopt** dintre care:
  - 12p daca implementarea obtine rezultate corecte
  - 3p pentru descrierea implementarii in README
- **20p** pentru implementarea variantei **opt_m** dintre care:
  - 15p daca implementarea obtine rezultate corecte si timpul de calcul pe partitia nehalem este mai mic de 14s pentru testul cu N = 1200
  - 5p pentru descrierea implementarii in README
- **9p** daca cele 3 implementari nu prezinta probleme de acces la memorie
  - Pentru a rezolva acest subpunct va trebui sa folositi **valgrind** cu optiunile **–tool=memcheck –leak-check=full**
  - Veti include 3 fisiere, **neopt.memory**, **blas.memory** si **opt_m.memory**, cu output-urile rularii valgrind pentru fiecare din cele 3 variante avand ca input fisierul **input_valgrind**
- **17p** pentru analiza celor 3 implementari folosind **cachegrind**
  - 6p pentru includerea in arhiva a 3 fisiere, **neopt.cache**, **blas.cache** si **opt_m.cache** reprezentand output-urile rularii **valgrind** cu optiunile **–tool=cachegrind –branch-sim=yes** pe partitia nehalem avand ca input fisierul **input_valgrind**
  - 6p pentru explicatii oferite despre valorile obtinute (I refs, D refs, Branches etc.)
  - 5p pentru explicatii oferite despre efectul optmizarilor facute de mana in varianta **opt_m** asupra valorilor obtinute
- **24p** pentru o analiza comparativa a performantei pentru cele 3 variante:
  - 15p pentru realizarea unor grafice relevante bazate pe rularea a cel putin 5 teste (5 valori diferite ale lui N: adica inca cel putin doua valori diferite de 400, 800 si 1200 pentru N)
  - 9p pentru explicatii oferite in README
- **(Bonus)**
  - Veti primi un bonus de maxim 10p daca timpul de calcul pentru varianta opt_m este mai mic de 9s pentru testul cu N = 1200
  - Consultati main.c pentru mai multe detalii
  - Bonusul se calculeaza doar pe partitia nehalem

Depunctări posibile:

- **neopt**
  - nu se tine cont de faptul ca A este matrice superior triunghiulara (intre -3p si -6p)
- **blas**:
  - nu se tine cont de faptul ca A este matrice superior triunghiulara (intre -3p si -6p)
  - unul sau mai multe calcule sunt realizate de mana, fara a folosi functii din BLAS (intre -3p si -15p)
  - a fost inclus codul BLAS (fisiere .so, .h., .c si altele) in arhiva temei (-15p)
- **opt_m**
  - nu se tine cont de faptul ca A este matrice superior triunghiulara (intre -3p si -6p)

        - inmultirea matricelor se realizeaza cu o complexitate diferita decat in cazul variantei neopt (ex. Strassen vs inmultire normala de matrice) (-15p)
        - timpul de calcul este mai mare decat timpul maxim permis - intre -5p si -15p
    - **analiza comparativa**
        - graficele nu au legenda / unitati de masura (intre -2p si -5p)
        - lipsesc partial sau complet timpii de rulare (intre -1p si -5p)
        - graficele nu contin toate datele cerute in enunt (intre -2p si -5p)
    - **generale**:
        - print-uri de debug in cod (intre -1p si -10p)
        - blocuri de cod comentate sau nefolosite (-1p)
        - warning-uri la compilare (intre -1p si -3p)
        - cod inghesuit/ilizibil (intre -1p si -3p)
        - implementare excesiva de functii in headere (-1p)
        - folosirea de constante hardcodate (-1p)
        - publicarea temei pe GitHub (-100p)

Precizari incarcare / VMChecker

Arhiva temei va fi incarcata pe vmchecker [https://vmchecker.cs.pub.ro/ui/#ASC].

Structura arhivei va fi urmatoarea:

```
src
  solver_blas.c
  solver_neopt.c
  solver_opt.c
  Makefile
  ...
cache
  blas.cache
  neopt.cache
  opt_m.cache
memory
  opt_m.memory
  neopt.memory
  blas.memory
README
grafice
  ...
```

La incarcarea pe vmchecker vor fi verificate urmatoarele:

- corectitudinea rezultatelor pentru cele 3 implementari
- lipsa problemelor de acces la memorie pentru cele 3 implementari
- prezenta unor fisiere .memory valide
- prezenta unor fisiere .cache valide

Celelalte aspecte ale temei (timpi limita pentru **opt_m** si bonus, README, grafice) vor fi verificate ulterior.

Precizări și recomandări

Timpul maxim pentru rularea celor 3 teste din fisierul **input** pe partitia nehalem folosind oricare din cele 3 variante este de 2 minute. Această limită de timp se referă la rularea întregului program, nu doar la partea intensiv computațională.

- Pentru a simplifica implementarea puteți presupune că N este multiplu de 40 și că este mai mic sau egal cu 1600.

- **În compararea rezultatelor se va permite o eroare absolută de maxim 10^{-3}.**
- În cazul variantei **opt_m** complexitatea trebuie să fie aceeași cu cea din varianta **neopt**.
- Formatul arhivei trebuie să fie **zip**.

Pentru a evita aglomerarea cozii se recomanda rularea de teste pentru valori ale lui N mai mici sau egale cu 1600.

Se recomandă ștergerea fișierelor coredump în cazul rulărilor care se termină cu eroare pentru a evita problemele cu spațiul de stocare.

În cazul în care job-urile vă rămân "agățate", va recomandam să utilizați de pe fep.grid.pub.ro, comanda

```
squeue
```

pentru a vedea câte job-uri aveți pornite, și apoi să utilizați comanda

```
scancel <job_id>
```

pentru a opri un job.

Resurse

- Ghid pentru folosirea gridului instituțional [https://infrastructure.pages.upb.ro/wiki/docs/grid-access]
- Schelet de cod