Tema 2

- Deadline soft: 15 aprilie 2019, ora 23:55
- Deadline hard: 22 aprilie 2019, ora 23:55

Enunț

Se dă următoarea operație cu matrice:

$$C = (zerotr(A^t \times B + B^t \times A))^2$$

unde:

- A si B sunt matrice patratice de double de dimensiune N x N
- A^t, respectiv B^t sunt transpusa lui A, respectiv transpusa lui B
- $\times$ este operația de înmulțire
- $+$ este operatia de adunare
- in urma aplicarii $zerotr(A)$ rezulta o matrice cu aceleasi dimensiuni ca matricea A si pentru care elementele de deasupra si de pe diagonala principala sunt cele din matricea A de pe pozitiile corespunzatoare, celelalte elemente fiind 0
- A^2 este A ridicat la patrat

Se dorește implementarea operației de mai sus in C/C++ în 4 moduri:

- **blas** - o variantă care folosește una sau mai multe functii din BLAS Atlas [1)]
- **neopt** - o variantă “de mână” fără îmbunătățiri
- **opt_m** - o variantă îmbunătățită a versiunii de mai sus. Îmbunătățirea are în vedere exclusiv modificarea codului pentru a obține performanțe mai bune
- **opt_f** - o altă variantă îmbunătățită obținută prin compilarea codului de la varianta **neopt** cu alte flag-uri de compilare ce aduc un bonus de performanță

Rulare și testare

Pentru testarea temei vă este oferit un schelet de cod pe care trebuie să-l completați cu implementarile pentru cele 4 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.
- **Makefile.icc** - Makefile folosit la compilarea cu icc.
- **input** - fișierul de input care contine 3 teste pentru urmatoarele valori ale lui N: 400, 1000, 1600
- **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** cu oricare din cele 2 Makefile-uri vor rezulta 4 fișere binare, **tema2_blas**, **tema2_neopt**, **tema2_opt_m** și **tema2_opt_f** corespunzătoare celor 4 variante care trebuie implementate.

Rularea se va realiza astfel:

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

unde:

- mod este unul din modurile **blas**, **neopt**, **opt_m** sau **opt_f**.
- 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 coada **ibm-dp.q**. Compilarea se va face folosind următoarele compilatoare:

- compiler/module **gcc-5.4.0** din modulul

  ```
  compilers/gnu-5.4.0
  ```

- Pentru a incarca modulul pentru GCC trebuie sa dati pe una din masinile din coada ibm-dp.q comanda

  ```
  module load compilers/gnu-5.4.0
  ```

- compiler/module **icc 16.0.2** din modulul

  ```
  utilities/intel parallel studio xe 2016
  ```

- Pentru a incarca modulul pentru Intel trebuie sa dati pe una din masinile din coada ibm-dp.q comanda

  ```
  module load utilities/intel_parallel_studio_xe_2016
  ```

Exceptând varianta **opt_f** nu se vor folosi flag-uri de compilare pentru optimizări.
Pentru linkarea cu BLAS Atlas se va folosi versiunea single-threaded **libsatlas.so.3.10** de pe masinile din coada ibm-dp.q
din

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

Fișierele output referință le găsiți aici:

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

Punctaj

Punctajul este impărțit 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** și **7.5p** pentru **icc**
- **10p** pentru descrierea implementarii pentru fiecare din cele 4 variante (**2.5p** pentru fiecare variantă)
- **30p** pentru analiza performantei dintre care:
  - **10p** pentru analiza comparativa **blas** vs **neopt** vs **opt_m** vs **opt_f** a performanței pentru **gcc 5p** și **icc 5p**
  - **10p** pentru o analiză comparativă **icc** vs **gcc**

  - **10p** pentru realizarea unor grafice relevante pe care să se bazeze analizele de mai sus
- **(Bonus)** 20p, respectiv 10p dacă timpul de rulare al variantei **opt_m** pentru ultimul test (N = 1600) este de cel mult 19 secunde, respectiv 23 de secunde, folosind unul din cele 2 compilatoare.

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

Precizări și recomandări

Timpul maxim pentru rularea celor 3 teste folosind oricare din cele 4 variante este de 4 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**.
- În cazul variantei **opt_f** se pot folosi atât flaguri de optimizare generice (O1/O2/O3) cât și flag-uri specifice ce nu sunt restricționate de folosirea unui anumit nivel de optimizare. [2) 3)]
- Formatul arhivei trebuie să fie **zip**.

Pentru analiza performantei se recomanda folosirea și altor date de intrare față de cele 3 teste din schelet.
Se recomandă ștergerea fișierelor coredump în cazul rulărilor care se termină cu eroare pentru a evita problemele cu spațiul de stocare.
Recomandăm folosirea comenzii

```
logout
```

după utilizarea sesiunilor interactive (cele în care intrați cu “qlogin -q ibm-dp.q”) și apoi să părăsiți și fep-ul tot cu **logout**.

În cazul în care sesiunile interactive (descrise mai sus) vă rămân “agățate”, să utilizați de pe fep.grid.pub.ro, comanda

```
 qstat
```

pentru a vedea câte sesiuni aveți pornite, și apoi să utilizați comanda

```
qdel -f <id-sesiune>
```

unde <id-sesiune> sunt primele cifre din stânga, rezultate după comanda **qstat**.

Resurse

- Cluster cheat sheet
- Schelet de cod

---

1) BLAS Atlas [http://www.netlib.org/blas/]

2) gcc - Optimize options [https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html]

3) Step by Step Performance Optimization with Intel® C++ Compiler [https://software.intel.com/en-us/articles/step-by-step-optimizing-with-intel-c-compiler]

tema2.txt · Last modified: 2019/04/09 12:59 by vlad.spoiala