**Accelerarea aplicării steganografiei**

**folosind CUDA**

1. Descrierea problemei

Steganografia reprezintă ascunderea unui mesaj, fișier, imagini sau sunet în interiorul altui mesaj, fișier, imagini sau sunet.

Tehnica a avut utilizări într-o multitudine de situații de-alungul istoriei, mai ales pentru transmiterea de mesaje ascunse. În era digitală, se poate automatiza procesul de ascundere a mesajelor și de decodare ulterioară a acestora.

Problema studiată în situația de față are în vedere ascunderea mesajelor text în interiorul imaginilor, mai exact ascunderea biților dintr-un mesaj ASCII pe biții cei mai puțin semnificativi ai pixelilor din imagini RGB.

2. Metode de paralelizare

Pentru a paraleliza problema de față, trebuie să luăm în considerare cele două categorii de algoritmi necesari de problemă: algoritmi care ascund mesajul și algoritmi care decodifică mesajul ascuns.

2.1 Metoda naivă

2.1.1 Algoritm de ascundere a mesajului

Pentru a paraleliza procesul de ascundere a mesajului, fiecărui thread i-am asociat un bit din mesaj și un pixel din imagine. Astfel, fiecare thread ascunde un bit din mesajul de codificat în imagine.

2.1.2 Algoritmi de decodificare a mesajului

Pentru a paraleliza procesul de decodificare a mesajului, fiecare thread poate lucra asupra unui pixel din imagine, sau asupra unui octet din mesaj.

Dacă un thread lucrează cu un singur pixel din imagine – respectiv un singur bit

din mesaj – apare o problemă de acces la memorie comună, întrucât locația de

memorie la care se scrie va fi accesată de câte 8 thread-uri.

Dacă un thread lucrează cu un singur octet din mesaj – respectiv 8 pixeli din

imagine – nu apare problema de acces la memorie partajată, dar performanțele vor fi mai reduse.

2.2 Metoda optimizată

2.2.1 Algoritm de ascundere a mesajului

Pentru ascunderea mesajului se caută pozițiile optime din imagine pentru a ascunde fișierul. În acest sens, biții din fișierul de ascuns vor fi inserați pe pozițiile 3:0 din pixelii imaginii, în acele locații în care variația pixelilor este cea mai mare, astfel încât informația să poată fi decodificată dar și greu observabilă.

2.2.2 Algoritm de extragere a mesajului

Pentru a extrage mesajul ascuns, se caută aceleași poziții în care variația între pixeli este mare, întrucât algorimtul de codificare nu modifică valoarea variației.

3. Resurse hardware

CPU: Intel I7 7700HQ @2.80 GHz

GPU: NVIDIA GeForce GTX 1050

OS: Windows 10, Student Edition

CUDA version: 10.1.243

4. Analiza rezultatelor

4.1 Varianta naivă de implementare

În urma măsurării timpului de execuție pentru codificarea mesajului, latențele sunt următoarele:

CPU: 0.003072ms

GPU kernels: 0.08000ms

GPU, including memory transfers: 0.519168ms

Se poate observa că pentru varianta neoptimizată a algoritmilor, codificarea durează mult mai mult pe GPU decât pe CPU. Setul de date utilizat a fost un string de dimensiune 2271 de octeți, inclusiv caracterul nul.

4.2 Varianta optimizată de implementare

Utilizând atât măsurători în cod, cât și cu Visual Profiler, am obținut următoarele rezultate:

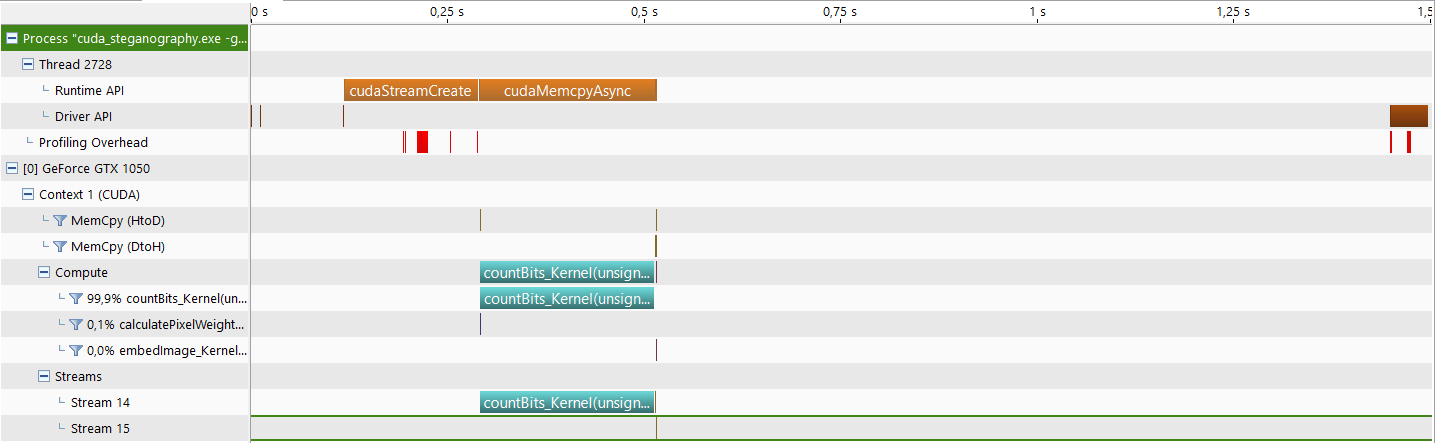


Figura 1 - Timpul de execuție pe GPU a operației de ascundere optimizată:

* 221.82244 ms pentru kernelul countBits\_Kernel;
* 42.018 us pentru kernelul embedImage\_Kernel;

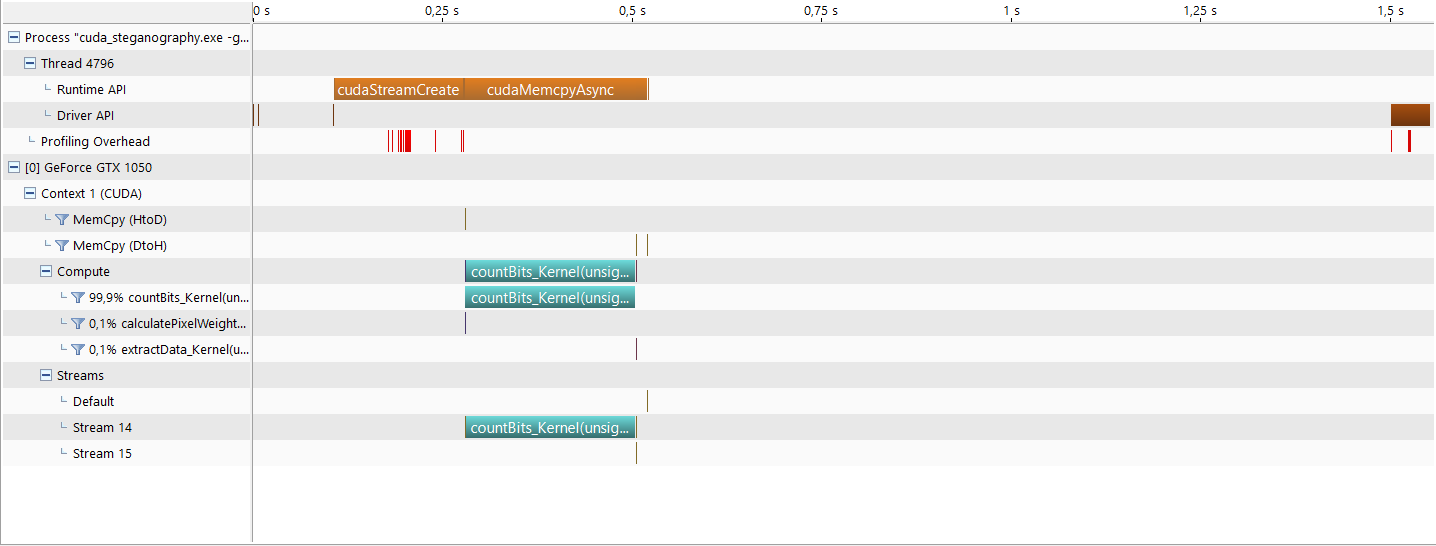


Figura 2 - Timpul de execuție pe GPU a operației de extragere optimizată:

* 224.58019 ms pentru kernelul countBits­\_Kernel;
* 117.989 us pentru kernelul extractData\_Kernel;

Pentru operațiile de pe CPU, timpii de execuție sunt:

* 18581 us pentru operația de ascundere
* 19564 us pentru operația de extragere

Se poate observa o latență uriașă impusă de transferul de date, chiar dacă s-au utilizat tehnici de optimizare. CUDA reușește să termine calculele pe fluxul de date înainte să se termine un transfer ulterior, motiv pentru care sistemul funcționează la cea viteza celei mai lente componente (PCIexpress).

Totuși, operațiile efective de codificare și decodificare a informației sunt foarte rapide (42us pentru codificare și 117us pentru decodificare).

5. Cod

<https://github.com/cristy-rotaru/cuda_steganography>