Ghent University

Computiationele biologie

Verslag project "FragGeneScan"

Tibo Vanheule

Academiejaar 2019-2020

Contents

1 Introductie

Er werd in de opgave drie keuzes voorgesteld:

- Bestaand fragScanGene source code optimaliseren en fout vrij maken.
- Bestaand optimalisaties van fragScanGene optimaliseren.
- FragSceneScan herschrijven.

Ik heb voor de eerste aanpak gekozen. Om deze code dus zo goed mogelijk te begrijpen, is Doxygen gebruikt [1]. Doxygen is vrij gemakkelijk om code te documenteren en te genereren als pdf of html. Het allereerste pdf document van deze generator is bijgvoegd **insert reference from zotero**. Via Javadoc-style comments kon ik gemakkelijk verder heel het project documenteren. De uiteindelijke documentatie is te vinden op https://tibovanheule.space/fraggenescan/ [2] en in het bijgevoegd pdf document **insert reference from zotero**.

2 Main

Als eerste werd de interface herschreven.

3 optimalisaties

3.1 Lees eerst, verwerk later

In de "get_train_from_file()" functie lezen we veel bestanden in. Het probleem is dat er lijn per lijn ingelezen wordt. Dit zorgt voor veel korte I/O operaties met elk een bijdragende vertraging. Om Dit te vermijden gaan we het bestand eerst volledig inladen in een buffer, zodat deze in reeds in het geheugen bevindt. De verwerking kan zo veel sneller gebeuren.

Memory map

4 code optimalisaties

In de code waren een heleboel code-optimalisatie mogelijkheden. In deze sectie worden de belangrijkste optimalisaties besproken.

4.1 Calloc

Op verschillende plaatsen in de code, maar vooral in util_lib **insert reference from zotero**. Wordt malloc gebruikt om geheugen te alloceren en dan wordt die waarden op nul gezet.

Calloc heeft dezelfde werking en uitkomst en vervangt deze 2 stappen door één instructie. Na een test **insert reference from zotero** ,bestaande uit 20000 iteraties, krijgen we volgende resultaten in figuur 1 (grafieken gemaakt in Power Bi [3])

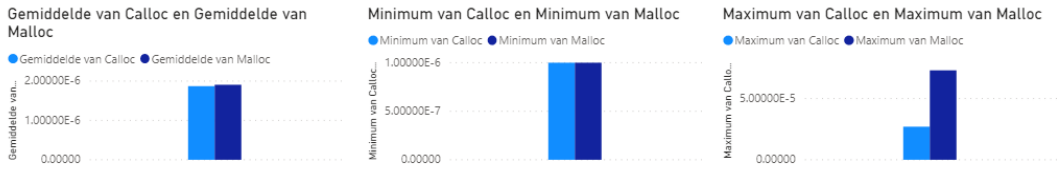

Figure 1: malloc vs calloc bij matrixen

In het aller slechtste geval is de calloc-instructie veel sneller dan de malloc-instructies. Er kan gesteld worden dat de calloc-instructie ofwel sneller is of even efficient is. Sinds calloc de code ook korter maakt en even leesbaar is als malloc, is op de mogelijke plaatsen malloc vervangen door calloc.

Figure 2: malloc vs calloc bij vectoren

4.2 Guards

Bij het openen van de header files bleek dat de conventies niet gevolgd waren. Een header file kon dus meerdere keren geincludeerd worden. Om problemen te vermijden werden guards toegevoegd [4].

References

[1] *Doxygen: Main Page*. URL: `http://www.doxygen.nl/`.

[2] *FragScanTibo: Hoofd Pagina*. URL: `https://tibovanheule.space/fraggenescan/`.

[3] *Gegevensvisualisatie? | Microsoft Power BI*. URL: `https://powerbi.microsoft.com/nl-nl/` (visited on 06/09/2020).

[4] *Include guard*. en. Page Version ID: 956136327. May 2020. URL: `https://en.wikipedia.org/w/index.php?title=Include_guard&oldid=956136327` (visited on 06/09/2020).