AALBORG UNIVERSITET
STUDENTERRAPPORT

Title:
C++ AMP og funcalc

Project period:
P10, forår 2016

Project Group:
?

Authors:
Niels Brøndum Pedersen
Supervisor:
Bent Thomsen

Total Pages: - 0
Appendix: - 1 CD
Completion date: - 00-00-0000

Synopsis:

NOT MADE

1 Intro

I tidligere Projekt blev der kigget på, om GPU'ens regnekraft kunne bruges i flere programmer for at øge ydeevne. I dette projekt var det også fundet, at det ikke var nemt at programmer til en GPU, fordi metoderne til at kode til en GPU ikke har en godt dokumentation og er kompliceret at lave. Derfor vil jeg i dette projekt prøve at at lave en simple programmering metode for at kunne bruge en GPU's regnekraft GPU, uden man skal havde de store viden inden for det. Regneark bliver brugt af mange og er forholdsvis nemt at programmere, derfor kunne det være interessant, at lave en funktion til et regneark program for at udregne med en GPU. Regneark programment jeg der blev brugt i dette projekt er *FuncalcFuncalc* der er open source. Inden jeg går i gang med at lave GPU funktion vil jeg kigge på tre API for at programmere til en GPU (CUDA, CUDAfy og C++ AMP), for at se hvad der virker godt i forhold til matrix Multiplatikon for at give et godt overblik over hvad der skal bruges når udvidelsen til regnearket skal laves.

Til at hjælpe med at fremstille resultater der kunne bruges, er Papiret *Microbenchmarks in Java and C#*[2] blevet brugt til at hjælpe med at fremstille resultater der kan bruges til at kigge på tid.

2 Test af Matrix Multiplatikon

For at give en bedre indblik på hvor meget speed-up det giver at bruge GPU'en i forhold til CPU'en, er der fremstillet simpel programmer der går en matrix multikapion. Der er lavet to versioner for hver GPU library, der er blevet kigget på. Forskellen mellem versionerne er, at det ene gør brug af en dimension og den anden bruger to dimensioner for array. De library der er blevet tested er C++ AMP, CUDA, for C++ og C# med normal CPU udregning er også lavet for kunne give en grundlag for hvor meget speed-up man for.

Testene bliver gjort for at give et bedre indblik om GPU er bedre en CPU, der er nogen forskel på hvordan input til GPU ser ud i forhold til om speed-up. Derefter er der blevet lavet et test program med C++ AMP der bliver kaldt fra C# kode, for at se hvordan dette kunne gøres og om det har den store effekt på udregnings tid med at bruge denne metode der er blevet fundet. Dette bliver gjort for fordi programmet Funcalc er skrevet i C#.

Testene er blevet fremstillet efter papiret (Microbenchmarks in Java and C#). Af de versioner af test papiret beskriver bruges der Mark3 version.

2.1 Resultater

CPU

CUDA

CUDAfy

C++ AMP

C++ AMP og C#

2.2 Efter Tanker

Det største problem jeg har haft med med CUDA og CUDAfy er, at man selv skal finde ud af hvor mange blokke og antal tråde pr. blok. Hvilket godt kan give nogen problemer når man arbejder med et program der skal arbejde med varierende input. Efter hvad jeg ar fundet på nettet angående problem med hvor mange blokke og antal tråde pr. blok man skal bruge har jeg for det meste fundet at man skal teste sig frem alt efter hvad der virker godt på det hardware man tester på.

For at løse dette problem for mig, har jeg lavet en generist metode der finder ud af hvor mange blokke der skal bruges, hvis det hele ikke kan gøres på en blok, denne metode er dog ikke perfekt.

En anden ting jeg er kommet på er plads mangle på GPU'en, min GPU kunne kun klare at gange matrixer der har max størrelse på ???. En løsning på dette problem kunne være at begynde at bruge billede hukommelse på GPU'en til readonly data. Hvilke skulle være muligt med CUDA og skulle ikke være muligt med CUDAfy.

En mindre fejl der begyndte at vise sig var at, når men kalder GPU funktion flere gange blev resultatet plusse med 3 hvis man ikke først sætter det til 0 inden man udregner. Denne fejl begundte at vise sig efter 3 udringer efter hinanden.

3 GPU-calculate til CoreCalc

I dette projekt vil CUDAfy blive brugt, til at øge regnekræften i open source programmet *Funcalc* der kan findes på hjemmesiden [1]. *Funcalc* en udvidelse til Corecalc, der er en implementering af et regneark funktionalitet lavet i sproget C#, det er lavet som et forskning prototype som ikke er ment for kunne blive brugt i stedet for de officielse versioner, såsom Microsoft Excel.

Klassen *GPU_ func* i GPU_ calculate mappen er hvor det meste arbejde ligger fra dette projekt. For at kunne bruge det jeg har fremstillet, er der også tilført noget kode i klassen *Function*.

3.1 GPU_ func

I *GPU_ func* er der seks funktioner og en konstruktør.

konstruktøren bliver brugt til at hente information om GPU'en der bruges til at bestemme om en blok er nok, hvis ikke hvor mange blokke skal der så bruges. Grunden til at information bliver hentet når man laver klassen er for minimere tiden funktion skal bruge på udregning, da jeg har observeret tager en god potion tid at hente denne information.

makeFunc og *makeFuncHelper* funktionerne bruges til at fremstille en liste med hvordan hvordan GPU'en skal udregne. Denne liste har *X* antal a fire fire tal som er

en enkle udregning (+,-,*,/). Tal første og tredje tal er hvad variabler der skal gøres noget med, det andet tal er for at bestemme hvilken udregning der skal gøres (+,-,*,/) og det fjerde og sidste bliver brugt til at bestemme om resultatet skal lige ligges i en midlertidig variable eller om den skal ligges i resultat listen.

For at give et eksemple kan vi tage regnestykket *(1+2)*(3+4)*, det her ses om en kommando for GPU funktion hvor tallene bliver brugt til at bestemme hvilken kolonne, den skal tag værdien fra. Den kunne komme til at se sådan ud: (1,1,2,-1),(3,1,4,-2),(-1,3,-2,0). 1+2 er blevet lavet om til (1,1,2,-1), 3+4 er blevet om til (3,1,4,-2) og ()*() er lavet til (-1,3,-2,0). Grunden til at der står minus -1 og -2 ved udregningen for 1+2 og 3+4, er at minus værdier bliver brugt til at beskrive midlertidig og 0 for ()*() bliver brugt til at beskrive at det skal i output listen.

3.2 Function

4 Test af GPU-calculate

4.1 Resultater

References

1. IT University of Copenhagen. Corecalc and funcalc spreadsheet technology in c sharp. `http://www.itu.dk/people/sestoft/funcalc/`.
2. Peter Sestoft (sestoft@itu.dk). Microbenchmarks in java and c sharp, 9 2015.