GPU Optimization of Base Form 820 Collective Artificial Spouse Retirement Plan

Niklas Schalck Johansson, Nikolaj Aaes and Hildur Flemberg
{nsjo, niaa, hufl}@itu.dk

IT University of Copenhagen

Abstract. Abstract

Keywords: CUDA, parallelism, threads, blocks, insurance, capital, software, programming languages, Runge Kutta 4th order

1 Introduction

- Why is the project relevant, what is the problem at hand
Scope? [2]
Hvad regner vi med at læseren har af kompetencer

This report is written at the IT University of Copenhagen in the spring term of 2013 in connection with a CUDA project supervised by Peter Sestoft from ITU and Hans Henrik Brandenborg Sørensen from DTU. The report is adressed to people interested in exploiting the benefits of using graphical processing units (GPUs) to optimise feasible computations in general, but specifically to people interested in CUDA.

In this project, we take a problem from the insurance industry that requires a significant amount of time to solve and show how to speed it up by moving the computations from the CPU to the GPU.

2 Background

- Related work, projects that precedes this project, why do we use the technology described, what is this new technology

2.1 Scope

SCOPE: Vi kører kun med een kunde ikke flere

2.2 Assumptions

lige nu er insurance holder altid en kvinde og spouse altid en mand en spouse er altid af det modsatte køn rentesatsen er altid 5 %

3 The Math

The algorithm in this project is used to determine the lump sum of money the insurance company needs to possess to be able to pay the insurance holder's spouse in the case of his or her death. The payment to the spouse will be in the form of a life interest which are disbursed from the time of death of the insurance holder unless the death occurs before the pension age. In that case the money will be disbursed after a grace period determined by the insurance company at the time the insurance was taken out.

We assume that the spouse of the insurance holder is of the opposite gender. If the insurance holder does not have a spouse at the time of his or her death the insurance is forfeited.

The algorithm used here is a 4th order Runge-Kutta solution with a fixed step-size (indsæt reference) where we use a series of constants determined before any calculation begins:

Constants

Name	**Meaning**
τ	The time of death of the insurance holder
r	The pension age
g	The grace period
x	The age of the insurance holder at calculation time ($t = 0$)
t	The time of calculation
h	The Stepsize of the Runge-Kutta solution

τ is expressed as $x + t$. Apart from this there is also a constant k which are determined by τ, r and g in the following manner:

How k is defined

If this statement holds	**then k equals**
$\tau < r$	g
$r \leq \tau < r + g$	$r + g - \tau$
$r + g \leq \tau$	0

The algorithm can be described as a combination of three models; an outer model that describes the life/death state of the insurance holder, a middle model that describes the married/unmarried state of the insurance holder and an inner model that describes the life/death state of the potential spouse.

NOTE: The description in the three model sections is loosely based on a describtion for Edlund[1] and some passages are directly translated from this document.

3.1 Outer model

The outer model is expressed as the following equation:

$\frac{d}{dt}f(t) = r(t)f(t) - \mu_t(x+t)(S^d_{x+t}(t) - f(t))$

Where $S^d_\tau$is the DØDSFALDSSUM that for a τ year old at the time t is needed to cover the payment from the insurance company, $\mu_t(x+t)$ is the mortality rate for a $(x+t)$ year old and $r(t)$] is the interest rate function. In this project the interest rate function returns the constant 0.05.

The differential equation is solved from $t = 120-x$ to $t = 0$ with the boundary condition $f(120-x) = 0$

3.2 Middle model

The middle model is used to calculate $S^d_{x+t}(t)$ and is expressed with the following equation:

$$S^d_\tau(t) = \begin{cases} g_\tau \int f(\eta|\tau) a^I_{[\eta]+g}(t)d\eta & \tau \leq r \\ g_\tau \int f(\eta|\tau) a^I_{[\eta]+r+g-\tau}(t)d\eta & r \leq \tau \leq r+g \\ g_\tau \int f(\eta|\tau) a^I_{[\eta]}(t)d\eta & r+g \leq \tau \end{cases}$$

Where g_τ is the propability that a τ year is married and $f(\eta|\tau)$ is the probability distribution for, that a τ year old is married to a η year provided that the τ year old is married. This equation can be rewritten to this form:

$\frac{d}{dn}f(\eta) = -g_\tau f(\eta|\tau) a^I_{[\eta]+k}(t)$

here the DET-DER-TAL-SOM-LIGNER-EN-EKSPONENT-MEN-ER-NEDE-I-STEDET-FOR-OPPE for a is rewritten to $[\eta]+k$ where k can fall into three different categories as shown in THE-TABLE-CONSTANTS.

This differential equation is solved from $\eta = 120$ to $\eta = 1$ with the boundary condition $f(120) = 0$

3.3 Inner model

The inner model is used to calculate $a^I_{[\eta]+k}(t)$ and is expressed with the following equation:

$\frac{d}{ds}f(s) = r(t+s)f(s) - 1_{s \geq k} - \mu_{t+s}(\eta+s)(0 - f(s))$

Where t, η and k are constants and $\mu_{t+s}(\eta+s)$ is the mortality rate for a $(\eta+s)$ year old.

This differential equation is solved from $s = 120-\eta$ to $s = 0$ with the boundary condition $f(120-\eta) = 0$

3.4 Runge-Kutta 4th order

The Runge-Kutta method is a method for approximating differential equations that builds on Eulers method (INDSÆT REFERENCE) and the midpoint method

(INDSÆT REFERENCE). When given a start point and a differential equation one can choose a stepsize and approximate the graph. When given a point (x_n, y_n), a differential equation f and a stepsize $h > 0$ one can use the Runge-Kutta method to approximate the next point in the following way:

$k1 = hf(x_n, y_n)$
$k2 = hf(x_n + \frac{h}{2}, y_n + \frac{k1}{h})$
$k3 = hf(x_n + \frac{h}{2}, y_n + \frac{k2}{h})$
$k4 = hf(x_n + h, y_n + k3)$

$x_{n+1} = x_n + h$
$y_{n+1} = y_n + \frac{k1}{6} + \frac{k2}{3} + \frac{k3}{3} + \frac{k4}{6} + O(h^5)$ HVORFOR KAN VI IGNORERE $O(h^5)$??

BESKRIV HVORDAN EN NORMAL KØRSEL SERUD = EKSEMPEL kig evt på en simpson integration når vi kigger på middle da det egentlig ikke er en differential ligning men et integrale

4 CUDA

en tråd skal lave et vist stykke arbejde før det kan betale sig at oprette den. Der er overhead for at lave context for hver tråd så hvis den arbejde tråden laver er mindre end at sætte overheaden op, giver det ingen mening.

- A more elaborate description of GPGPUs, architecture and CUDA C.

4.1 Parralellisation

De forskellige parallelliserings muligheder: - beskriv teorien - beskriv hvad vi får ud af det i køretid? - beskriv hvorfor vi har/ikke har valgt denne løsning - Kan den kombineres med de andre for bedre performance?

Outer Par
Da man på forhånd kan beregne alle x værdier outer skal bruge i sin kørsel og alle Middle(x) kald i outerDiff kun skal bruge en x værdi er det muligt at regne alle disse middle kald i parallel og gemme hvert middle kalds resultat i et array. Når man derefter skal gå fra startpunkt til næste punkt kan man betragte Middle(p.x) i $returnr_(px) * py - GmFemale(x + px) * (Middle(px).y - py)$; som en udregning der tager konstant tid.

Middle Par
Da man på forhånd kan beregne alle x værdier middle skal bruge i sin kørsel og vi ved at hvert step i Middle har udregninger til k1,k2/k3 og k4 i hvert skridt der kun afhænger af x-værdien kan vi regne alle de sæt af k1,k2/k3 og k4 som middle skal bruge i parallel. Derefter kan de lægges sammen som et vægtet gennemsnit i et array og bruger når man vil gå fra et punkt til et nyt. Kaldet $doubley = p.y + k1/6 + k2/3 + k3/3 + k4/6$; kan altså betragtes som konstant.

Her udregnes hvert sæt af k1, k2/k3 og k4 dog i en enkelt tråd. Det vil sige at der er een tråd for hvert step.

Inner Par
Man har også muligheden for at parallisere middles kald til inner der bruges i udregningen af k1, k2/k3 og k4. Ved at sætte 3 tråde til at udregne de tre værdier kan dette gå tre gange så hurtigt.

Kombination af Outer Par og Middle par
Ved en valgt block size b = antal outersteps og en thread per block tpb kan man sætte hver block til at tage sig af en x-værdi i outer, altså tage sig af et enkelt middle kald. Hvis man forestiller sig at middle skal tage ms steps og ms > tpb, vil man først udregne k1, k2/k3 og k4 sæt for de først tpb steps og derefter gå videre til de til næste tpb steps indtil alle steps er udregnet. Det betyder også at der kan være inaktive tråde i den sidste omgang tpb steps og det kan give divergence. Det er muligt at bruge reduction (se cuda bogen) når alle k er blevet regnet ud og man skal lægge p.y til.

Kombination af Middle par og Inner par
Ligesom i den ovenstående løsning lader man hver block handle en x-værdi for middle og giv hver block tre tråde hver af disse tråde udregner hver sin k værdi og hver tredje tråd lægger dem sammen.

Kombination af alle tre
Som i kombinationen af outer par og middle oar vælger man et antal blokke svarende til antallet af outersteps og en thread per block tpb. Man lader stadig hver blok tage sig af et enkelt kald til Middle med en bestemt x-værdi. Forskellen kommer når man i stedet for at lade hvert tråd udregne et sæt af k1, k2/k3 og k4 værdier sætter man i stedet 3 tråde til at udregne dem i parallel. Dette vil kun have indvirkning hvis block størrelsen er større end det antal steps middle skal udføre. Hvis ikke bruger man tre gange så mange tråde til at så udregnes først tpb k-værdier, tre gange hurtigere end normal, dernæst de næste tpb k-værdier, stadig tre gange så hurtigt men i stedet for at stoppe ved middlesteps som normal vil man stoppe ved middlesteps*3 da der skal bruges 3 gange så mange k-værdier. Optimalt set skal der altså være middlesteps*3 tråde i hvert block. Ved en vilkårlig stepsize vil det være 120*stepsize*3 tråde. Med en stepsize på 1 eller to giver det hendholdsvis 360 og 720 tråde hvilket stadig er inden for det 1024-tråde limit hver block har. Men allerede ved stepsize 3 giver det 1080 tråde per block hvilket ikke er legalt.

Hvad har vi valgt
Vi har valgt Kombination af Outer Par og Middle Par for simpelheden. Kombination af alle tre er ikke smart fordi det kun giver optimering når threads per block er middlesteps*3 eller over hvilket ikke er legalt med en stepsize over 2. SKRIV NOGET OM HVORFOR VI HAR VALGT KOMBI AF OUTER OG

MIDDLE I STEDET FOR MIDDLE OG INNER. HVORFOR ER DET MERE EFFEKTIVIT?

5 Implementation

- A description of the new implementation and design choices I den kode der er nu er insurance holder kvinde og spouse mand ALTID skriv noget om memory bound og compute bound i rapporten

6 Testing

test på forskellige maskiner hvis vi deler steps op i blocks af 32 tråde kan der i worst case scenario være en sidste block med 1 aktiv og 31 inaktive tråde. Lav en graf over rest trådene

- A test to ensure that the new implementation produces the same output as the current implementation

7 Benchmarks and Comparison

- Benchmarking and comparison on speed between the current and new implementation

8 Conclusion

Conclusion

9 Reflection / Discussion / Future improvements

improvements: det kan optimeres til at kunne køre flere kunder. Simpson løsning i middle?

KUNNE DET VÆRE INTERESSANT AT BRUGE DYNAMIC PROGRAMMING? KUNNE MAN GEMME HVILKE KALD TIL INNER DER ER LAVET OG HVIS INNER SÅ KALDES IGEN MED SAMME VÆRDIER, KAN MAN SLIPPE FOR AT REGNE DET UD OG BARE HENTE RESULTATET. DOG SKAL MAN VÆRE OPMÆRKSOM PÅ AT MAN KAN KOMME I EN SITUATION HVOR EN BEREGNING SER AT DER IKKE ER NOGET RESULTAT FOR ET SÆT PARAMETRE OG DERFOR GÅR I GANG MED AT REGNE DET UD, MENS DEN FØRST BEREGNING KØRER KAN ANDRE BEREGNINGER MED SAMME PARAMETRE OG DE NYE BEREGNINGER VIL OGSÅ GERNE GEMME RESULTATET NÅR DE ER FÆRDIGE HVILKET KAN GIVE CONFLICTS (MÅSKE). eller Det kan være vi skal dokumentere en del om at man kan gemme alle resultater i en tabel og efter et vist antal

kunder kan man ende med nok til bare at have opslag i stedet for beregninger. skal denne tabel et ligge på GPU så den kan slås op i konstant tid.

skriv evt noget om autotuning, hvor programmet først tester med forskellige mængder af tråde og blokke og vælger den hurtigste løsning.

- Did we achieve what we wanted? what did we discover during the project? What can be changed in future implementations? Threats to validity??? kan det betale sig at køre dele af programmet på CPU'en?

10 Glossary

- Forsikringstager - insurance holder
- ægtefælle - Spouse
- livrente - life interest
- dødfaldssum
- pause periode - grace period
- randbetingelse - boundary condition

References

[1] EDLUND DOKUMENTET
[2] David B. Kirk, Wen-mei W. Hwu - Programming Massively Parallel Proccesors, A Hands-on Approach - Elsevier Inc. - 2010