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

Table of Contents

Abstract. Abstract is the executive summary of the entire paper. As a summary it should encompass not only the main result, but also the problem, and the evaluation. Shaw [4] reports that good ICSE papers discussed evaluation already in their abstract. This is important in software engineering, because of the emphasis put on the evaluation of results. A good abstract is readable for non-experts, as it increases the chances that someone will build on your work (often application opportunities appear in other areas). I try to have four sentences in my abstract. The first states the problem. The second states why the problem is a problem. The third is my startling sentence. The fourth states the implication of my startling sentence.
Simon Peyton Jones summarizes this idea in the following points
Statetheproblem Saywhyit'saninterestingproblem Saywhatyoursolutionachieves Saywhatfollowsfromyoursolution

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

1 Introduction

This report is written at the IT University of Copenhagen (ITU) in the spring term of 2013 in connection with a CUDA project supervised by Peter Sestoft from ITU and Hans Henrik Brandenborg Sørensen from the Technical University of Denmark (DTU). The report is addressed to people interested in exploiting the benefits of using general purpose graphical processing units (GPGPUs)[1] to optimize 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 using the GPU. The problem concerns life insurance policies for married people, where one of them receives an amount of money some time after the other person has died. The goal, from the view of the insurance company, is to estimate its holding as accurately as possible to always be able to satisfy the obligation to the policyholder. Such an estimate is largely dependent on a guess of when the policyholder is going to die which can be anytime between signing of the policy and some, possibly large, number of years ahead in time. Furthermore, the time of death is dependent on several variables like age, gender etc.

The insurance company has a mathematical model that unifies all these variables and can be solved by a computer but it takes a lot of time to do - even with

[1]Graphical Processing Units (GPUs) are most commonly used to render graphics in computer games. GPGPUs are "general purpose" variants that share the same architecture as GPUs but are optimized for computing purposes out of line with the graphics category. Since this project is not about graphics, we will refer to GPGPUs simply as GPUs, for shortness, throughout the rest of the report.

modern CPUs. Moreover, insurance companies typically issue several thousands of policies, each of which has an impact on the complexity of the computation as well as the size of the overall reserves the company needs to hold back in case a disbursement is claimed. All these facts together make computing power a valuable entity. The goal of this project is to analyze the opportunity to speed up the computation for a single policyholder through parallelism with CUDA, i.e. to optimize the time it takes to compute the size of the reserves needed at any time during the life of a policyholder such that the insurance company can fulfill its obligations when the person dies.

The reader of the report is assumed to know the basic principles of concurrency and parallelism. No CUDA specific knowledge is required.

2 Background

- Related work, projects that precedes this project, why do we use the technology described, what is this new technology - Nævn at vi har fået udleveret matematik beforehand

2.1 Problem definition

How can CUDA C be utilized to optimize the calculation of the baseform 820 collective artificial spouse retirement plan using the Runge Kutta 4th order method?

2.2 Scope

The scope of this project is to handle a single customer at a time. We do not explore the possibilities for optimizing the process of using the algorithm on several customers at a time. Neither do we explore alternative methods for approximating differential equations.

2.3 Assumptions

This project was developed under several assumptions to limit the size of it. We assumed that the spouse of an insurance holder was always of the opposite gender. It would now be possible to take same-gender couples into account but it would require som modification to the code. We assumed that the function to determine the interest rate always returns 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 stepsize (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 death benefit 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 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 Implementation

- A description of the new implementation and design choices

Husk at nævne at Middle stepsize er hardcoded til 2. I den kode der er nu er insurance holder kvinde og spouse mand ALTID

5 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.

5.1 CUDA Architecture

5.2 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?

6 Testing

To ensure that the program produced the desired result we chose to run a series of executions with the C# implementation and our CUDA implementation and compare the results. We ran INDSÆT TAL HER test and INDSÆT TAL HER of them was successful in producing identical results. INDSÆT TAL HER was not.

The test were conducted on INDSÆT TAL HER machines with the following specifications:
INDSÆT SPECIFIKATIONER HER

lige nu er insurance holder altid en kvinde og spouse altid en mand

Vi har lavet X antal kørsler der alle giver det samme resultat. Læg dem i appendix eller som filer på CD'en

7 Benchmarks and Comparison

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. skriv noget om memory bound og compute bound i rapporten - Benchmarking and comparison on speed between the current and new implementation

Tre slags grafer:
køretid og stepsize med konstante g,r,x. Tag ca 9 stikprøver.
CUDA køretid og rest tråde.
køretid og variable g,r,x.

8 Reflection / Discussion

improvements: Simpson løsning i middle?

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

Threats to validity??? kan det betale sig at køre dele af programmet på CPU'en?

- Did we achieve what we wanted? what did we discover during the project? What can be changed in future implementations?

When computing several customers one after another it could be beneficial to store the result of each calculation of the outer, middle and inner model and what parameters they were given. Whenever the program wanted to calculate value it could check if the calculation was already performed at a previous time and instantly get the result instead of calculating it again. If the result was not found in the storage it could calculate it itself and store the result afterwards for future use.

This approach is only beneficial when the lookup in the storage is faster than performing the calculation itself. It is also not beneficial if the amount of calculations that are identical, is not large enough. It would be interesting to examine the calculation collision with a large amount of executions with different customers. A drawback with this approach is that the storage can become very large over time and that could potentially lead to slower lookups. Since the approach stops being beneficial when the lookups are slower than the actual calculations it would be smart to examine when this occurs and evaluate what results is stored. If it occurs too fast one might consider to only store the middle or outer results because they are slower to calculate than the inner results.

9 Future work

This project has a limited scope and we would like to document suggestions for future expansion of the project:

Formal verification of correctness of code
We assume that the code produces the correct results because it is modeled after existing code and because our test of INDSÆT KORREKT TAL values are identical to the results produced by the existing code. Our test give a sense of correctness but without formal verification there is always the possibility that a corner case is not covered and will produce incorrect results.

Handling multiple customers
This project is built around the assumption that it would be run for a single customer at a time. If it is to have any real world use it must be able to run multiple customers at the same time. While this is possible with the current implementation is has not been tested or optimized. It would be interesting to examine how GPGPUs could be utilized when handling multiple customers and if the time used per customer could be even lower than it is when handling a single customer.

Variable interest curve
In this project the interest rate is assumed to be a constant of 5 %. However in the real world this is not the case. The interest rate is variable over time and the program should be able to take this into account.

Switch genders at runtime

When handling multiple customers in the same execution being able to switch the genders of the insurance holder and spouse at runtime is necessary. This is not a complex extension of the program but it is still something that is not implemented in this project. The difference between using female and male is the function used to calculate their mortality intensities. Changing whether the female or male function should be called for each customer and spouse should be trivial in the C and C# implementation but might have repercussions in the CUDA C implementation since it can generate branch divergence.

Same-gender marriages

As an addition to switching gender at runtime the program will be able to emulate the real world better if it supported same-gender marriages. The same considerations are present as the function to calculate mortality intensities is now the same for both the insurance holder and the spouse. Additionally the function used to calculate the probability that the insurance holder is married might be different for same-gender marriages.

10 Conclusion

Conclusion

11 Glossary

- Forsikringstager - insurance holder
- ægtefælle - Spouse
- livrente - life interest
- dødfaldssum - Death benefit
- 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