Using High Performance Computing for Optimizing Credit Risk Calculation

Mark Joselli
Jose Ricardo Silva Junior
Marcelo Zamith
Esteban Clua
MediaLab, IC-UFF

Eduardo Soluri
Nullpointer Tecnologia

Abstract

The volume of banks data calculation is increasing each year with extraordinary scale and with that, new forms of computation is needed. High performance computing is a very attractive field for optimization such bank calculous, which can give promising results. This paper shows a implementation of know model for assessing the credit risk of a company. For getting most accurate price and speedup comparisson, this method was implemented in both CPU and GPU version. The Gpu version was builtt using CUDA architecture and show some reasons and advantages of using such the Gpu computing for computational finance.

Keywords:: High performance computing, NVidia, CUDA, GPU Computing, finance, Merton Model, Credit Risk

Author's Contact:

{mjoselli, jricardo, mzamith, esteban}@ic.uff.br
esoluri@nullpointer.com.br

1 Introduction

In the finance market, in the last few years, firms and banks are searching for solutions which could improve the performance of its calculation of a high volume of data. Also this processing need not to consume high financial power to buy and maintain data centers.

Credit risk arises whenever a borrower is expecting to use the cash flow of the future to pay a debt in the present. Investors are compensated for assuming this risk and the higher the perceived credit risk, the higher the rate of interest that investors will demand for lending their capital. Credit risks are calculated based on the borrowers' overall ability to repay, and there are diverse methods for this calculation. These models takes into account the borrowers' collateral assets, revenue-generating ability and taxing authority (such as for government and municipal bonds). Credit risks are normally a vital component of fixed-income investing.

The Merton model is a model proposed in 1974 by Robert C. Merton [Merton 1973] for assessing the credit risk of a company. This models characterizes the company's equity as a call option on its own assets. This model is built on the Black and Scholes (1973) [Black and Scholes 1973] and Merton (1973) framework to find the value of the debt issued by the firm. The ideas were already in Black and Scholes REF, who discussed the view of the firm as a call option.

The Merton model is empirically accurate for non-financial firms, especially manufacturing entities, as REF shows. For financial firms this model is not recommended due to its highly leveraged nature which produces CDS spreads.

The development of programmable GPUs has opened new possibilities for general-purpose computation (GPGPU) or Gpu computing, which can be used, for instance, to enhance the level of realism of virtual simulations. Some examples of works in GPGPU that address these issues are Quantum Monte Carlos [Anderson et al. 2007], finite state machines [Rudomn et al. 2005], games REF and ray casting [Muller et al. 2007].

2 GPGPU

GPUs are powerful processors originally dedicated to graphics computation. It is composed by several parallel processors, allowing it to present much better performance then modern CPUs in several applications scenarios. An nVidia 8800 ultra [NVIDIA 2006],for instance, can sustain a measured 384 GFLOPS/s against 35.3 GFLOPS/s for the 2.6 Ghz dual-core Intel Xeon 5150 [NVIDIA 2008].

The GPUs architectures are being specially designed for processing tasks that require high arithmetic rates and data bandwidths. Because of the SIMD parallel architecture of the GPU, the development of this kind of application requires a different programming paradigm than the traditional CPU sequential programming model (the nVidia GeForce 9800 GX2 [nVidia 2009b], for example, has 256 unified stream processors.). In order to take advantage of the GPU processing power in a game, the developer needs to adapt the game tasks to this kind of parallel paradigm, like the architecture present in this work does.

nVidia and AMD/ATI are implementing unified architectures in their GPUs. Each architecture is associated with a specific language: nVidia has developed CUDA (Compute Unified Architecture) [nVidia 2009a] and AMD developed CAL (Compute Abstraction Layer) [AMD 2008]. One main advantage in the use of these languages is that they allow the use of the GPU in a more flexible way (both languages are based on the C language) without some of the traditional shader languages limitations (such as scatter memory operations, i.e. indexed write array operations), and offering others features that are not even implemented on those languages (such as integer data operands like bit-wise logical operations AND, OR, XOR, NOT and bit-shifts) [Owens et al. 2007].

Nevertheless, the disadvantage of these software architectures is that they are only available for the hardware of the proper vendors, i.e., CUDA only works on Nvidia and CAL only works on AMD/ATI cards. In order to have GPGPU programs that work on both GPUs it is necessary to implement them in shader languages like GLSL (OpenGL Shading Language), HLSL (High Level Shader Language) or CG (C for Graphics) with all the vertex and pixel shader limitations and idiosyncrasies. According to its vendors, in the near future it will be possible to use OpenCL (Open Computing Language) [Group 2009], which is available for both nVidia and AMD graphics cards.

The GPGPU is getting more and more common and it is being applied in many fields, like geologic [Kadlec et al. 2009], medical [Muyan-Ozcelik et al. 2008] and computer vision [TunaCode 2010]. The websites from CUDA [nVidia 2010] and gpgpu.org [GPGPU.org 2010] shows the latest development in the field.

3 Gpu computing in financial compute

The optimization of diverse mathematical algorithms used in the financial marked with the use of Gpu computing has show as a promising research field. Big corporate financial institutions have dedicating projects and resource in order to search for improvements of its solution or the development of new solutions using the Gpu architecture.

We can see the use of Gpu computing on the prediction of options in many works. The work [Cvetanoska and Stojanovski 2012] shows on the american option using an back-sholes method and [Tse

et al. 2011] show on the asian option with Monte-Carlo method. Solomon et al.[Solomon et al. 2010] presents a trinomial lattice strategy for the pricing of simple European options on the GPU. Surkov [Surkov 2010] shows an implementation on both the european and american option with Fourier space time-stepping on GPU. Also [Gaikwad and Toke 2010] uses the Gpu for solving multidimensional option pricing PDEs using sparse grid combination technique.

Prediction of the market is a promising field, and Gpu could also be used of its optimization. With that the GPU could be used for simulating in real time the Value-at-Risk (VaR) estimation to measure the severity of portfolios losses. The work [Dixon et al. 2009] shows delta-gamma Value-at-Risk implementation on the GPU with expressive speedup. Also similar methods could be use for evaluation of portfolio data of user of the bank, in order to suggest options and stocks for them.

In the balance of a large portfolio of stocks, analysts need to search for short-term and long-term patterns, identify correlations between securities, and also develop forecasts. This problem, which is also a VaR for Hedge Fund Trading require intensive computations against large databases, of more than a decade of trading data, of thousands of securities. Using GPU algorithm would allow a faster reaction time to market conditions, and also enabling the evaluation more sophisticated algorithms that take into account larger data sets. The work from Dixon et al [Dixon et al. 2009] show promises results on the field.

Detection of Credit Card Fraud algorithms like [Duman and Ozcelik 2011; Ju and Wang 2009] could also be used in Gpu, but there are lack of works on this field using Gpu computing for processing. This problems comes form the rise of identity tefh together with the popularity of online shopping has resulted in a huge increase in credit card fraud. The thieves are becoming increasingly shrewd in exploiting security weaknesses with that, banks and credit card companies need to be fast to detect the frauds. With the GPU computing a bank could run a more sophisticated fraud detection algorithms against millions of credit card accounts.

Also the credit risk is a promising field. The book [Hwu 2011] and the paper [Thomas and Luk 2008] shows a work on GPU that uses monte-carlo as a model. This work show the same problem, with a different model for solving, the Merton method, which is shown in the next section.

4 The Merton Model

The literature on credit risk starts with the paper by Merton (1974) [Merton 1973], which presents the Merton Model. This models applies the option pricing theory developed by Black and Scholes (1973) [Black and Scholes 1973] to the modelling of a rms debt [Elizalde 2006].

This model assumes that the capital structure of the firm is comprised by equity and by a zero-coupon bond (also called a discount bond) with a maturity T and face value of D, whose values at time t are denoted by E_t and $z(t,T)$ respectively, for $0 \leq t \leq T$. The firms asset value V_t is calculated by summing the equity and debt values of the firm.

With this, the equity represents the call option of the firms assets, also with a maturity of T and strike price of D. The firm can only default at time T, this assumption is important to be able to treat the firms equity as a vanilla European call option, and therefore apply the Black-Scholes pricing formula, and then two cases can happens:

- The firm does not default (which occurs when a firm has not met his legal obligations according to the debt contract) and shareholders receive $V_t - D$ if at maturity T the firms asset value V_t is enough to pay back the face value of the debt D.
- Otherwise, ($V_t < D$) the firm defaults, bondholders take control of the firm, and shareholders receive nothing.

The rms asset value is assumed to follow a di?usion process given by:

$$dV_t = rV_t dt + \sigma_V V_t dW_t \tag{1}$$

where σ_V is the (relative) asset volatility and W_t is a Brownian motion.

The payo?s to equityholders and bondholders at the time T are respectively, $max(V_t?D, 0)$ and $V_t?E_t$, i.e.

$$E_T = max(V_T?D, 0), \tag{2}$$

$$z(T,T) = V_T?E_T. \tag{3}$$

Applying the Black-Scholes pricing formula, the value of equity at time $t(0 \leq t \leq T)$ is given by

$$E_t(V_t, \sigma_V, T-t) = e^{-r(T-t)}[e^{r(T-t)}V_t\gamma(d_1) - D\gamma(d_2)] \tag{4}$$

where γ is the distribution function of a standard normal random variable and d_1 and d_2 are given by

$$d_1 = \frac{\ln(\frac{e^{r(T-t)}V_t}{D}) + frac12\sigma_V^2(T-t)}{\sigma_V\sqrt{T-t}} \tag{5}$$

$$d_2 = d_1 - \sigma_V\sqrt{T-t} \tag{6}$$

Where the probability of default at time T is given by

$$P[VT < D] = \gamma(?d2). \tag{7}$$

Therefore, the value of the debt at time t is $z(t,T) = V_t - E_t$.

In order to implement this model, the rms asset values of both the V_t and its volatility σ_V needs to be estimate. Also the debt of the rm has to be transformed into a zero-coupon bond with maturity T and face value D.

5 The Implementation

Our implementation of the Merton's model for credit risk calculation was based on the sequential CPU code, which came from the book [Press et al. 1992]. This code was adapted and further optimized in order to run on CPU and on GPU. The code was implemented using the language C and for Gpu computing the CUDA was used.

For value input of the variables of the problem, the assets price, volatility, the maturity and the face variable are based on real case scenarios. Also random number are used in order to gather a more massive data, where values based on the real one are calculated.

The sequential version implements double precision floating numbers, while the Gpu version implements single precision floating numbers. The L1 Norm of the difference between CPU and GPU results is monitored for every credit risk calculation, in order to control the precision.

5.1 Results

For the tests, we used a PC with a Ubuntu 10.10 Linux distribution equipped with an Intel i7 3.07GHz using 8 GB of RAM and a NVidia GeForce GTX 580 which has 512 cores. Simulations tests with different configurations were performed. To assure the results are consistent, each test was repeated 10 times and the standard deviation of the average times confirmed to be within 5%.

Table 1: *Numerical results, in milliseconds, from the new physics loop implemented on the GPU and CPU*

# Firms	CPU Time	GPU time	Speedup
65536	29.790	0.683	43.62
262144	119.19	2.519	47.32
1048576	477.0299	8.231	57.95
4194304	1905.240	27.038	70.46

From these results, can be seen that the GPU is much faster that the CPU, even when only 1000 firms are calculated. As the number of firms increases, so the speedup of the GPU compared to the CPU.

Credit Risk using Merton's model is can be easily divided among multiples Gpus, this is because the calculation of the risk of each firm is independent of all others. Therefore the computation can be distributed across multiple GPUs present in the system. To provide parallelism across multiple GPUs, the set of input firms is divided into contiguous subsets, which the number of subsets equals the number of CUDA-capable GPUs installed in the system, and then they are passed to host threads driving individual GPU contexts. The API state is encapsulated inside a Gpu context, so there is always a one-to-one correspondence between host threads and gnu contexts.

6 Conclusion

This article shows how the GPU can be used to calculate credit risk using Merton model more efficiently than the CPU. As the GPUs continue to become faster more quickly than CPUs, the difference in performance between the two is likely to grow for this type of application.

We showed that the parallel algorithm implemented on the GPU is more than 6500 times faster than the serial one implemented on the CPU.

Also the code developed could be adapted to be used in multiples Gpus, in order to gain scale.

The authors of this work will continue on researching compute financial with the use of Gpu computing, with the davelopment of new problems of the field.

References

AMD, 2008. Amd stream computing. Avalible at: http://ati.amd.com/technology/streamcomputing/firestream-sdk-whitepaper.pdf. 20/02/2008.

ANDERSON, A., III, W. G., AND SCHRDER, P. 2007. Quantum monte carlo on graphical processing units. *Computer Physics Communications 177(3)*.

BLACK, F., AND SCHOLES, M. 1973. The Pricing of Options and Corporate Liabilities. *The Journal of Political Economy 81*, 3, 637–654.

CVETANOSKA, V., AND STOJANOVSKI, T. 2012. Using high performance computing and monte carlo simulation for pricing american options. Tech. Rep. arXiv:1205.0106, May. Comments: CIIT Conference, April 2012, Bitola Macedonia.

DIXON, M., CHONG, J., AND KEUTZER, K. 2009. Acceleration of market value-at-risk estimation. In *Proceedings of the 2nd Workshop on High Performance Computational Finance*, ACM, New York, NY, USA, WHPCF '09, 5:1–5:8.

DUMAN, E., AND OZCELIK, M. H. 2011. Detecting credit card fraud by genetic algorithm and scatter search. *Expert Syst. Appl. 38*, 10 (Sept.), 13057–13063.

ELIZALDE, A. 2006. Credit risk models ii: Structural models. Tech. rep., CEMFI.

GAIKWAD, A., AND TOKE, I. M. 2010. Parallel iterative linear solvers on gpu: A financial engineering case. In *Proceedings of the 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, IEEE Computer Society, Washington, DC, USA, PDP '10, 607–614.

GPGPU.ORG, 2010. Gpgpu.org. Avalible at: http://www.gpgpu.org.

GROUP, K., 2009. Opencl - the open standard for parallel programming of heterogeneous systems. Avalible at: http://www.khronos.org/opencl/.

HWU, W. 2011. *GPU Computing Gems Jade Edition*. Applications of GPU Computing Series. Elsevier Science.

JU, C., AND WANG, N. 2009. Research on credit card fraud detection model based on similar coefficient sum. In *Proceedings of the 2009 First International Workshop on Database Technology and Applications*, IEEE Computer Society, Washington, DC, USA, DBTA '09, 295–298.

KADLEC, B., TUFO, H., AND DORN, G. 2009. Knowledge-assisted visualization and segmentation of geologic features using implicit surfaces. *IEEE Computer Graphics and Applications 99*, PrePrints.

MERTON, R. C. 1973. On the pricing of corporate debt: the risk structure of interest rates. Working papers 684-73., Massachusetts Institute of Technology (MIT), Sloan School of Management.

MULLER, C., STRENGERT, M., AND ERTL, T. 2007. Adaptive load balancing for raycasting of non-uniformly bricked volumes. *Parallel Computing 33(6)*, 406–419.

MUYAN-OZCELIK, P., OWENS, J. D., XIA, J., AND SAMANT, S. S. 2008. Fast deformable registration on the gpu: A cuda implementation of demons. In *the 1st technical session on UnConventional High Performance Computing (UCHPC) in conjunction with the 6th International Conference on Computational Science and Its Applications (ICCSA)*, IEEE Computer Society, Los Alamitos, California, M. Gavrilova, O. Gervasi, A. Lagan, Y. Mun, and A. Iglesias, Eds., ICCSA 2008, 223–233.

NVIDIA. 2006. Geforce 8800 gpu architecture overview. tb-02787-001_v0.9. Technical report, NVIDIA.

NVIDIA. 2008. Nvidia - cuda compute unified device architecture. Programming guide, NVIDIA.

NVIDIA, 2009. Nvidia cuda compute unified device architecture documentation version 2.2. Avalible at: http://developer.nvidia.com/object/cuda.html.

NVIDIA, 2009. nvidia geforce 9800 gx2 specification. Avalible at: http://www.nvidia.com/object/product_geforce_9800_gx2_us.html.

NVIDIA, 2010. Cuda zone. Avalible at: http://www.nvidia.com/object/cuda_home_new.html.

OWENS, J. D., LEUBKE, D., GOVINDARAJU, N., HARRIS, M., KRGER, J., LEFOHN, A. E., AND PURCELL, T. J. 2007. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum 26(1)*, 80–113.

PRESS, W., TEUKOLSKY, S., VETTERLING, W., AND FLANNERY, B. 1992. *Numerical Recipes in C*, 2nd ed. Cambridge University Press, Cambridge, UK.

RUDOMN, T., MILLN, E., AND HERNNDEZ, B. 2005. Fragment shaders for agent animation using finite state machines. *Simulation Modelling Practice and Theory 13(8)*, 741–751.

SOLOMON, S., THULASIRAM, R. K., AND THULASIRAMAN, P. 2010. Option pricing on the gpu. In *Proceedings of the 2010 IEEE 12th International Conference on High Performance Computing and Communications*, IEEE Computer Society, Washington, DC, USA, HPCC '10, 289–296.

SURKOV, V. 2010. Parallel option pricing with fourier space time-stepping method on graphics processing units. *Parallel Comput. 36*, 7 (July), 372–380.

THOMAS, D. B., AND LUK, W. 2008. Credit risk modelling using hardware accelerated monte-carlo simulation. In *Proc. IEEE Symp. on FPGAs for Custom Computing Machines*.

TSE, A. H., THOMAS, D. B., TSOI, K. H., AND LUK, W. 2011. Efficient reconfigurable design for pricing asian options. *SIGARCH Comput. Archit. News 38*, 4 (Jan.), 14–20.

TUNACODE, 2010. Cuvilib: Cuda vision and imaging library. Avalible at: http://www.cuvilib.com/.