Team Deku (1)

Code link

https://github.com/professorcode1/IntelHack

This is team Dekus submission for the problem

> Parallelizing Monte Carlo Simulation of Financial Derivatives using SYCL

Currently the team composes of only one member. I have experience coding GPU's and using Monte Carlo methods.

Introduction

The problem states that a monte carlo simulation must be used in the derivation of financial derivatives using data and SYCL must be used to implement said solution. There are plenty of papers that cover the usage of Monte Carlo Simulation to derive financial derivates[1-8]. However I focus our attention on 3 resources [9-11]. This is because these methods use probabilistic machine learning to tackle the problem of stock prediction which is better because

1) it is trivially parallelisable as apposed to quasi monte carlo methods like [3] and LSTM like [7] so they are better suited for GPU and FGPA and will be able to utilise the power of SYCL

2) They are not black boxes as opposed to method like Neural Networks

3) And expert knowledge can be easily incorporated. How is explained below

The disadvantages of the methods id

1. The models discussed in all the papers completely ignore all data such as Open, High Low Close, etc. All they use is Daily Returns.

I will attempt to tackle this disadvantage if I get enough time

Mathematical overview of the purposed solution

All mathematics modelling of stock prices start from the base fact that Stock price is a stochastic variable (unlike say velocity from physics which is deterministic) and is given by the function

$$dS_t = \mu S_t dt + \sigma S_t dB_t$$

where $\mu S_t dt$ is the drift and $S_t dB_t$ is volatility where B_t is a Brownian motion. This has an analytical solution

$$S_t = S_0 e^{(\mu - \frac{\sigma^2}{2})t + \sigma B_t}$$

Where the exponent term is the called the “Return”. Or

$$Return = \ln(\frac{S_t}{S_{t-1}})$$

Stock obeys geometric brownian motion. Now there are many method that attempt to find $\mu\ and\ \sigma$ at some time t given historical data. I will use Hull and White [11] which is

$$\theta = U(0,1)$$
$$\mu = N(-5,1)$$
$$\sigma = U(0.001,2)$$
$$dW = U(0,0.1)$$
$$y_t = y_{t-1} + \theta.(\mu - t_{t-1}) + \sigma.dW$$
$$Return = Normal(0, y_t)$$

The distribution of the parameters θ, μ, σ, dW are what have been observed in the stock market throughout time.

Equations like of the form that y_t is in are solved using Euler-Maruyama algorithm[12] where the result is a posterior function. To collapse a posterior function to a solution, I use a sampling method for eg Hastings Algorithm which is a algorithm belonging to the class of algorithms called **Markonikov Chain Monte Carlo (MCMC).** The entire class of MCMC relies of constructing a Markonikov chain using Monte Carlo sampling to collapse a prior function to a posterior function.

Expert Knowledge

All Baysiam Methods really on the simple principle. Given a prior distribution for our random variable(observable and internal), a model of how those random variable are related, and some data over observed variables return the improved probability distribution of the internal random variables.

Say you have a software company from the India which codes blockchain for its clients and it IPO's a few days ago. You have very little data on this company, however there has been another company, also from India which also gives this exact service that IPO'd years ago and on that you have years of data. By using that to create your prior, you would get much better results than if you used all the stock data, not only that but the processing required will also reduce.

Expert knowledge says that when creating the posterior, use only those data that are similar, and by giving different number of training loops to different training data, you can prioritise some data over other on the level of how similar you think they are.

Implementation

There will be 2 singleton classes called DataManagement and SYCLComputer.

Data class

This will also be a singleton which will connected to a database of stocks. Given expert knowledge (tags) it will go through the data and return a list of (dataset, importance). This class will be responsible for loading data from disk to memory. All data will be managed by this class whether it be on RAM,V-RAM or disk. Unified Shared Memory hence will not be used(not for any other reason than me preferring to always make it explicit where my data is). This however is also an advantage for if in the future data becomes too large this class can effectively be used to schedule data around so the model doesn’t have to fail execution all together.

```
class DataManagement{
private:
DatabaseManagement()
float calculateImportance();
public:
static DatabaseManagementInstance;
vector<pair<vector<dataset>, float>> fetch(expertKnowledge);
}
```

SYCLComputer

All SYCL code will be written as private methods to class and the public methods in this class will be used to implement the algorithm.

Now the actual algorithm will be implemented as a method to the SYCLComputer class. The class will have 1 queue for host and a list of queue’s for devices. For now there are no plans for a CPU cluster. This method will have multiple iterations starting from simplest to most optimised. This is to prevent the possibility of the hack ending without a single working prototype being produced.

Below is the psuedo code

```
const State state0 = initState()
const State[] states = []
while(true){
  samples = MetropolisHastings(state0)
  y = EulerMaruyama(samples, states)
  State state(y);
```

```
   states.push(state)
 }
```

The problem is immediately evident. Euler-Maruyama is a serial algorithm, we needs to iterate over the solutions from the previous iteration and is hence serial in nature

Project Iterations

I will iterate over the solution by stating with the slowest and easiest implementation and moving up to the most optimised implementation. This is to ensure that the hackathon doesn’t end without a single working solution.

Iteration 1

The entire algorithm is coded uses the default queue that SYCL provides. Just pure functions and USMs. No DataManagement class, no kernels.

Iteration 2

The MetropolisHastings algorithm has a kernel written for it since it is trivially parallelisable(because interanlly it uses monte carlo sampling which is trivially parallelisable). However data will have to be shifted back and forth b/w CPU and GPU constantly since EulerMaruyama is serial hence is best run on CPU.

Iteration 3

The EulerMaruyama is shifted to the GPU as well. It is serial in the exact same way same that LSTM’s are serial. And Nvidia parallelised them here.[13]. This paper is used as reference to implement the kernel that executes EulerMaruyama on the GPU.

Iteration 4

The model is optimised using Intel advisor and DataManagement class is created(hence UAT’s are dropped). DataManagement class is important since it isn’t possible to keep all required data in VRAM. MetropolisHastings is replaced with more state of the art MCMC algorithms like NUTS(No U-Turn Sampler)

References

[1] Does Price Limit Affect the Autocorrelation of Stock Return Series? A Monte Carlo Experiment

[2]**Monte Carlo Simulation Prediction of Stock Prices:**10.1109/DeSE54285.2021.9719349

[3]Parallel and Distributed Computing Issues in Pricing Financial Derivatives through Quasi Monte Carlo:Ashok Srinivasan

[4]Using Least-Square Monte Carlo Simulation to Price American Multi Underlying Stock Options: Irma Palupi, Indra Utama Sitorus, Rian Febrian Umbara

[5]Stock Market Valuation using Monte Carlo Simulation: Sehba Shahabuddin Siddiqui, Vandana A. Patil

[6] Proposed System for Estimating Intrinsic Value of Stock Using Monte Carlo Simulation:Sehba Shahabuddin Siddiqui, Vandana A. Patil

[7]Study on the Risk-Return Mathematical Model Based on LSTM Time-Series Model and Monte Carlo Simulations:Wenqi Zhou, YaqianGuo,Sha Li

[8]SEO Pricing with Marketability Restriction. A Monte Carlo Method with Stochastic Return and Volatility: Xu Zhaoyu, An Shi

[9] Brodd, Tobias and Adrian Djerf. “Monte Carlo Simulations of Stock Prices : Modelling the probability of future stock returns.” (2018).

[10] M. Hariadi, A. A. Muhammad and S. M. S. Nugroho, "Prediction of Stock Prices Using Markov Chain Monte Carlo," 2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM), Surabaya, Indonesia, 2020, pp. 385-390, doi: 10.1109/CENIM51130.2020.9297965.

[11]J. Hull and A. White, The pricing of options on assets with stochastic volatilities, Journal of Finance 42 (1987), 281–300

[12]https://en.wikipedia.org/wiki/Euler–Maruyama_method

[13] **Optimizing Performance of Recurrent Neural Networks on GPUs:** https://arxiv.org/abs/1604.01946