Accelerating Option Risk Analytics in R Using GPUs

Matthew Dixon[1], Sabbir Khan[2] and Mohammad Zubair[2]

[1]Department of Analytics
School of Management
University of San Francisco
Email: mfdixon@usfca.edu
[2]Department of Computer Science
Old Dominion University
Norfolk, VA
Email: zubair@cs.odu.edu

GTC'14
27th March, 2014

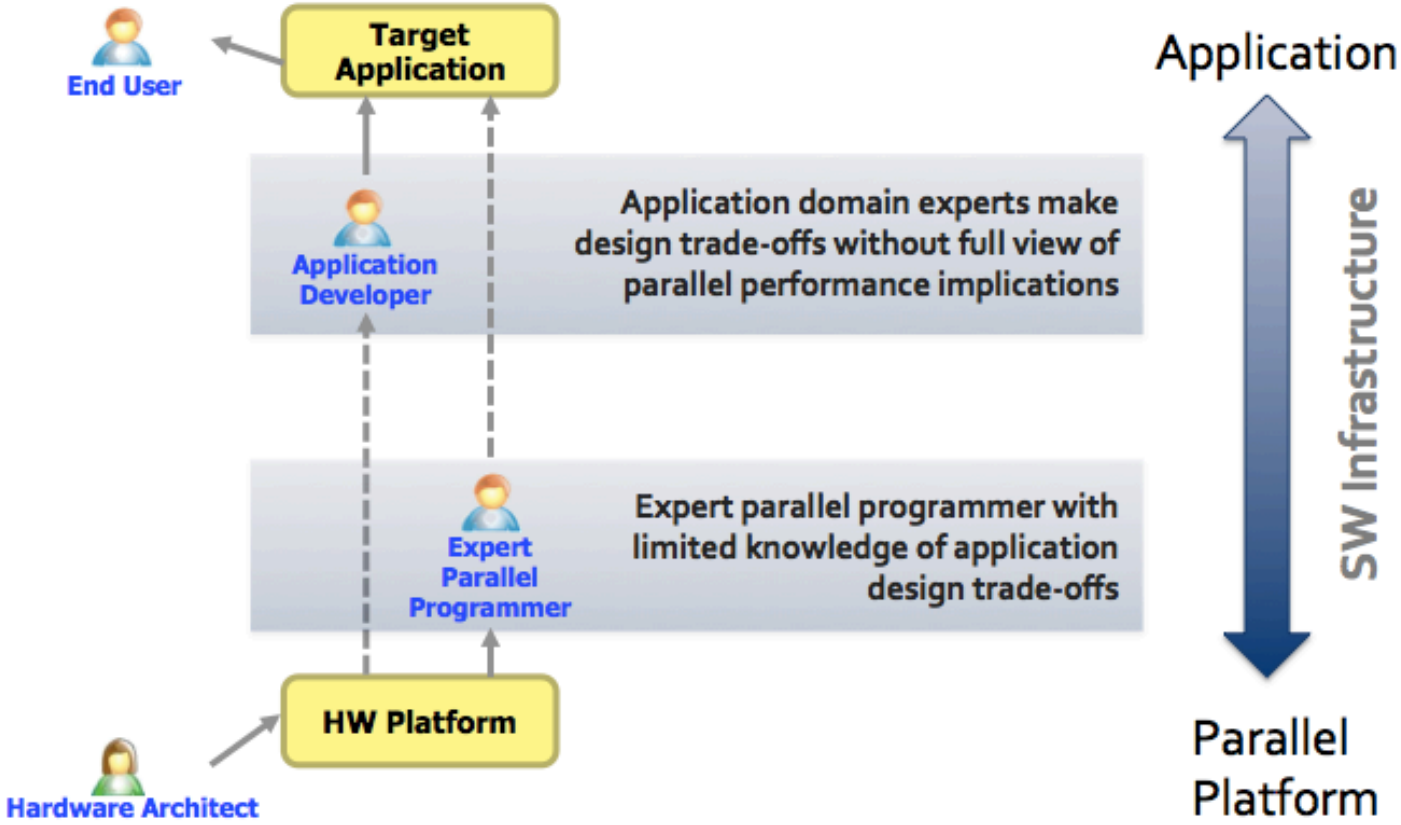
Background resources

- Paper:
  M.F. Dixon, S. Khan and M. Zubair, gpusvcalibration: A R Package for Fast Stochastic Volatility Model Calibration using GPUs. http://ssrn.com/abstract=2394306.
- Development version of R package:
  https://github.com/mfrdixon/gpusvcalibration

Overview of presentation

- Learn why stochastic volatility (SV) models are used extensively across the capital markets for pricing and risk management of exchange traded financial options.
- Understand some of the factors which make calibration of SV models computationally intensive
- Gain insight into the pros and cons of using the R statistical software environment for calibrating SV models
- Learn about the gpusvcalibration R package for accelerating stochastic volatility model calibration on GPUs, which provides a factor of up to 760x performance improvement on a NVIDIA Tesla K20c (Kepler architecture) consisting of 2496 cores

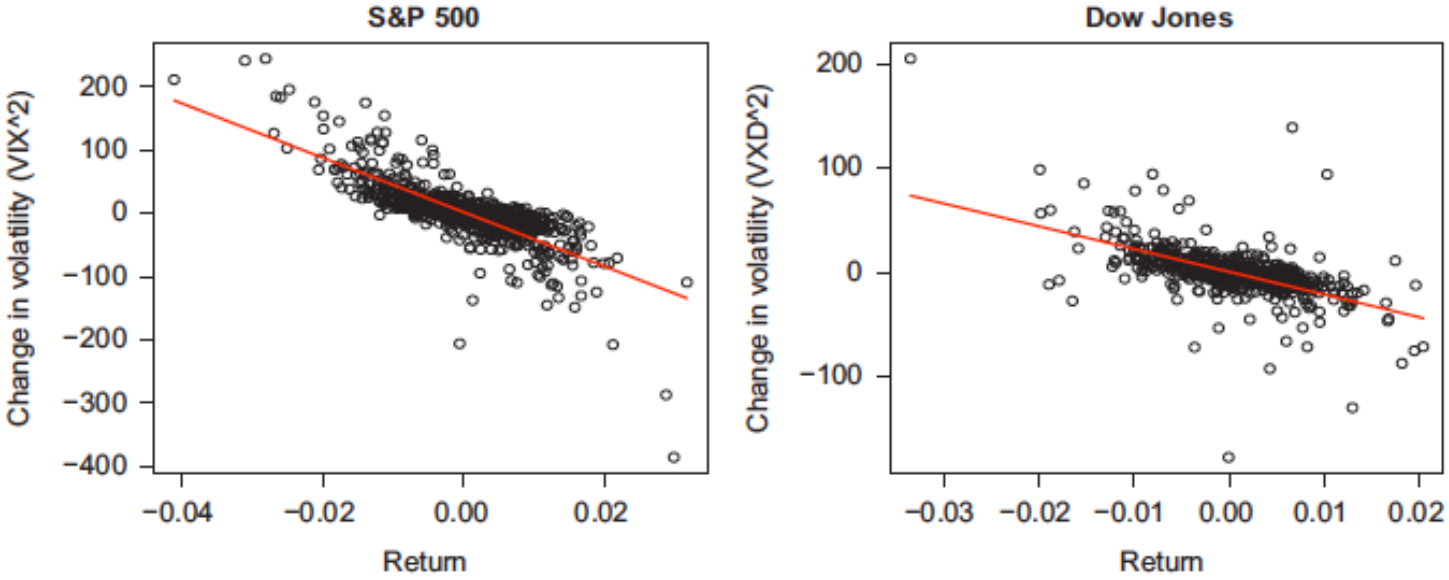
Implementation Gap



Background on Stochastic Volatility Modeling

1. Market makers quote options for strike-expiry pairs which are illiquid or not listed
2. Pricing engines, which are used to price exotic options and which are based on far more realistic assumptions than Black-Scholes-Merton model, are calibrated against an observed Implied Volatility Surface (IVS)
3. The IVS given by a listed market serves as the market of primary hedging instruments against volatility and gamma risk (second-order sensitivity with respect to the spot)
4. Risk managers use stress scenarios defined on the IVS to visualize and quantify the risk inherent to option portfolios.

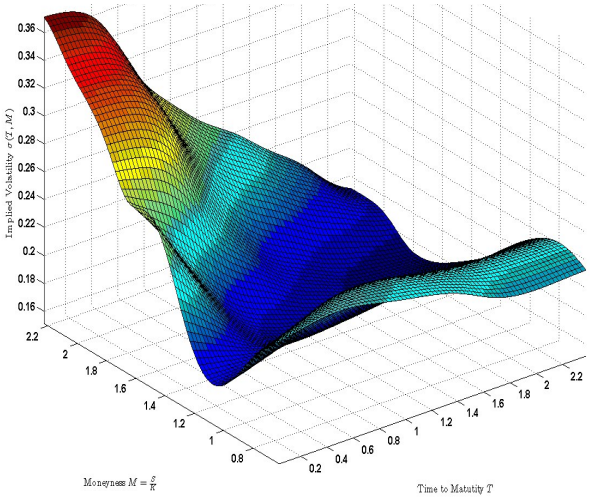
Evidence of the Leverage Effect



Daily changes of squared volatility indices versus daily returns. Using the volatility indices as the proxy of volatility, the leverage effect can clearly be seen. Left: S&P 500 data from January 2004 to December 2007, in which the VIX is used as a proxy of the volatility; Right: Dow Jones Industrial Average data from January 2005 to March 2007 in which the Chicago Board Options Exchange (CBOE) DJIA Volatility Index (VXD) is used as the volatility measure.

Source: Yacine Ait-Sahalia, Jianqing Fan and Yingying Li, The leverage effect puzzle: Disentangling sources of bias at high frequency, Journal of Financial Economics 109 (2013): 224–249

Stochastic Volatility Modeling



3. Imply the volatility surface from the fitted model prices

4. Trading desks use the implied volatility surface to price non-quoted options and price and hedge exotics

1. Choose a SV model, e.g. Bates Model

Definition (Bates Model)

$$\frac{dS_t}{S_t} = \mu dt + \sqrt{V_t} dW_t^1 + (Y-1) S_t dN_t, \quad (1)$$

$$\frac{dV_t}{V_t} = \kappa(\theta - V_t)dt + \sigma\sqrt{V_t} dW_t^2, \quad (2)$$

- *V_t is given by a mean reverting square root process constrained by $2\kappa\theta - \sigma^2 > 0$.*
- *N_t is a standard Poisson process with intensity $\lambda > 0$ and*
- *Y is the log-normal jump size distribution, where $\mu_j = ln(1+a) - \frac{\sigma_j^2}{2}$, $a > -1$ and $\sigma_j \geq 0$.*

2. Estimate the European Option Price

Definition (Bates European Option Pricing Model)

- *Call price of a vanilla European option is*

$$C(S_0, K, \tau; \mathbf{p}) = S_0 P_1 - K \exp\{-(r-q)\tau\} P_2,$$

- *P_1 and P_2 can be expressed as:*

$$P_j = \frac{1}{2} + \frac{1}{\pi} \int_o^\infty Re\left[\frac{\exp\{-iulnK\}\phi_j(S_0, \tau, u; \mathbf{p})}{iu}\right] du, j = 1, 2.$$

- *where the parameter set $\mathbf{p} := [\theta, \sigma, \kappa, \rho, v_0, \mu_j, \sigma_j, \lambda]$.*

Error convergence of FFT versus Fourier-Cosine[1]

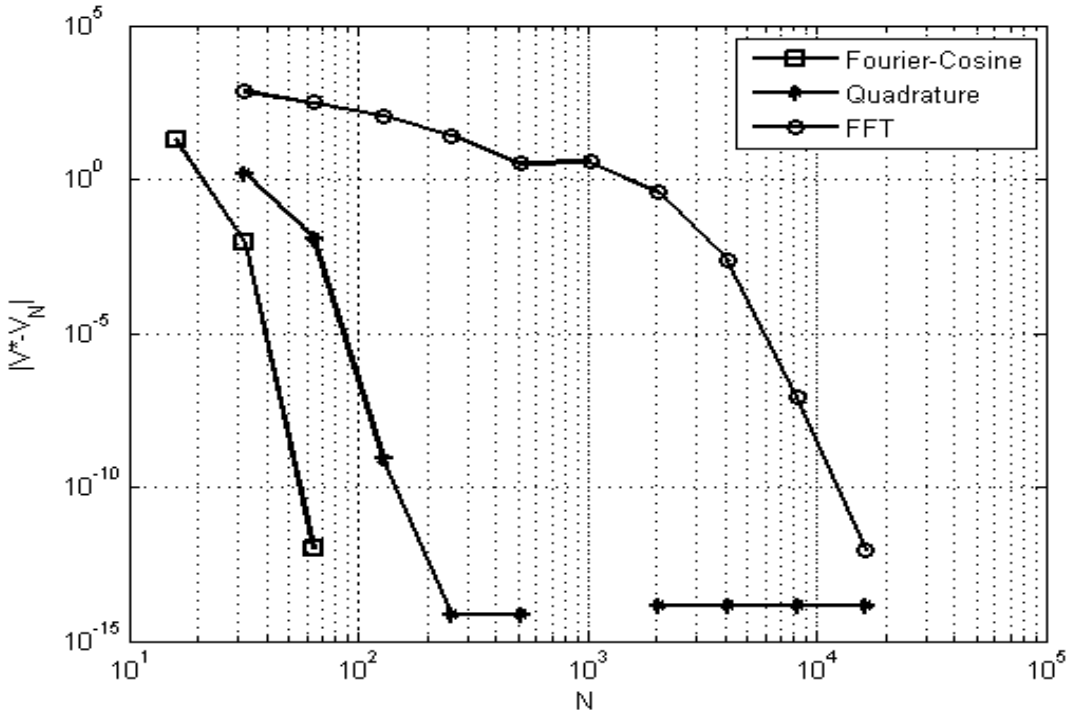


Figure : Comparison of the error convergence rates of the Fourier-Cosine, fixed second order Gauss-Legendre quadrature and Carr-Madan FFT methods applied to the Heston pricing model.

[1] F. Fang, C.W. Oosterlee, SIAM Journal on Scientific Computing 31 (2), 2008, 826-848

There is a need to refrequently re-calibrate the models

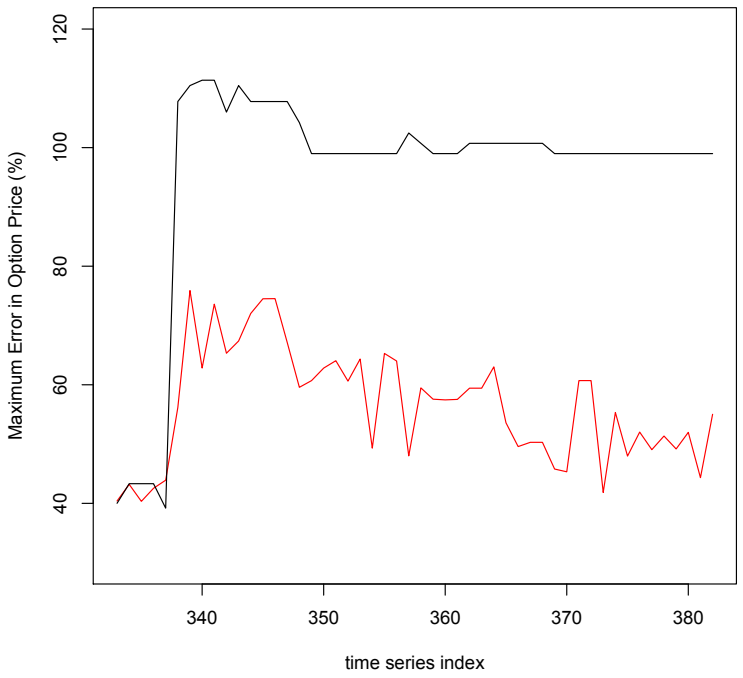


Figure : The maximum point-wise error across the volatility surface for options on ZNGA against time. The red-line shows the error resulting from calibration of the Bates model every 30 seconds versus calibrating at the start of the period (black-line).

Global calibration in R

1. Call the Differential Evolution Algorithm - this is a stochastic parallel direct search evolution strategy optimization method. Specify a maximum number of candidates and a tolerance on the error function. The DE algorithm is available in the *DEoptim* package[2].
2. Call an iterative constraint based optimizer: specify tolerances on various errors, e.g. relative function error or relative solution changes. The *NLoptr* package provides various local optimizers.
3. *Performance penalty using R*: can not quickly calibrate the model which hinders modeling, testing and productionization.

[2] D. Ardia, J. David, O. Arango and N.D.G. Gomez, Jump-Diffusion Calibration using Differential Evolution, Willmott Magazine, 55 Sep, 76-79, 2011.

Chain data

- For calibrating the option price model we consider a sample chain of n option data $ch[n]$, where the ith chain data has the following properties:

$ch[i].u$:	Underlying asset price
$ch[i].s$:	Strike price
$ch[i].m$:	Time to maturity
$ch[i].p$:	Option price

Sequential ErrorFunction(p)

```
rmse ← 0
for i = 0 to n − 1 do
    vp ← V(ch[i].u, ch[i].s, ch[i].m, p)
    diff ← ch[i].p − vp
    rmse ← rmse + diff × diff
end for
rmse ← SQRT(rmse/n)
return rmse
```

Sequential implementation of the calibration program in R

	AAPL	AMZN	BP	CSCO	GOOG	MSFT
DEoptim	293	111	69	62	255	70
nloptr	148	56	35	32	130	36
ErrorFunction	1.46	0.55	0.34	0.31	1.28	0.35
Total ErrorFunction	440	166	103	93	385	105
Total Time	441	167	104	94	386	106
% ErrorFunction	99.8%	99.4%	99.1%	99.0%	99.7%	99.1%

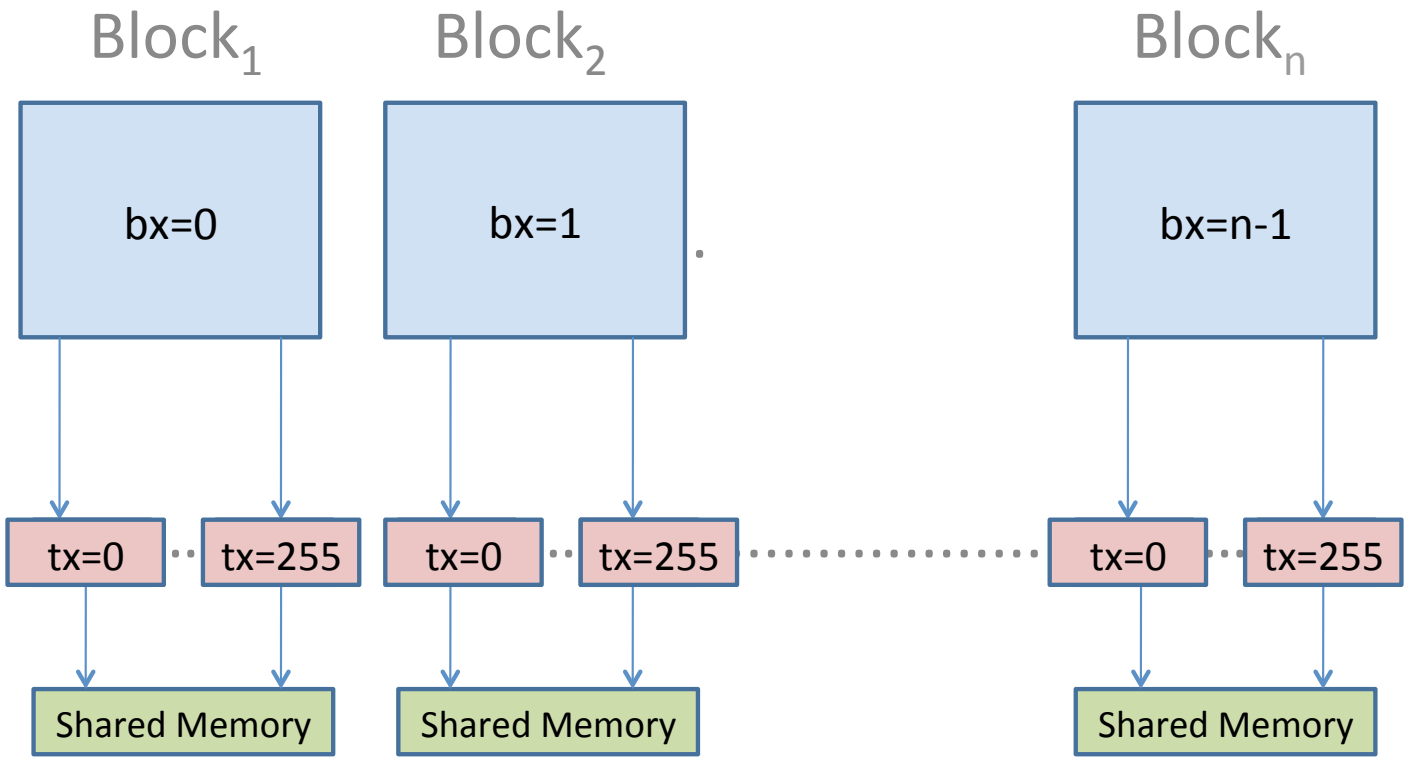
Table : Performance results for the R code in seconds. Each columns represents a different option chain.

Sequential implementation of the calibration program in C

	AAPL	AMZN	BP	CSCO	GOOG	MSFT
DEoptim	88	33	21	18	76	22
Nlopt	44	17	10	9	38	11
ErrorFunction	0.44	0.17	0.1	0.1	0.38	0.11
Total ErrorFunction	131	49	30	27.6	113	32
Total Time	132	50	31	27	114	33
% ErrorFunction	99.2%	98.0%	96.8%	99.9%	99.1%	97.0%

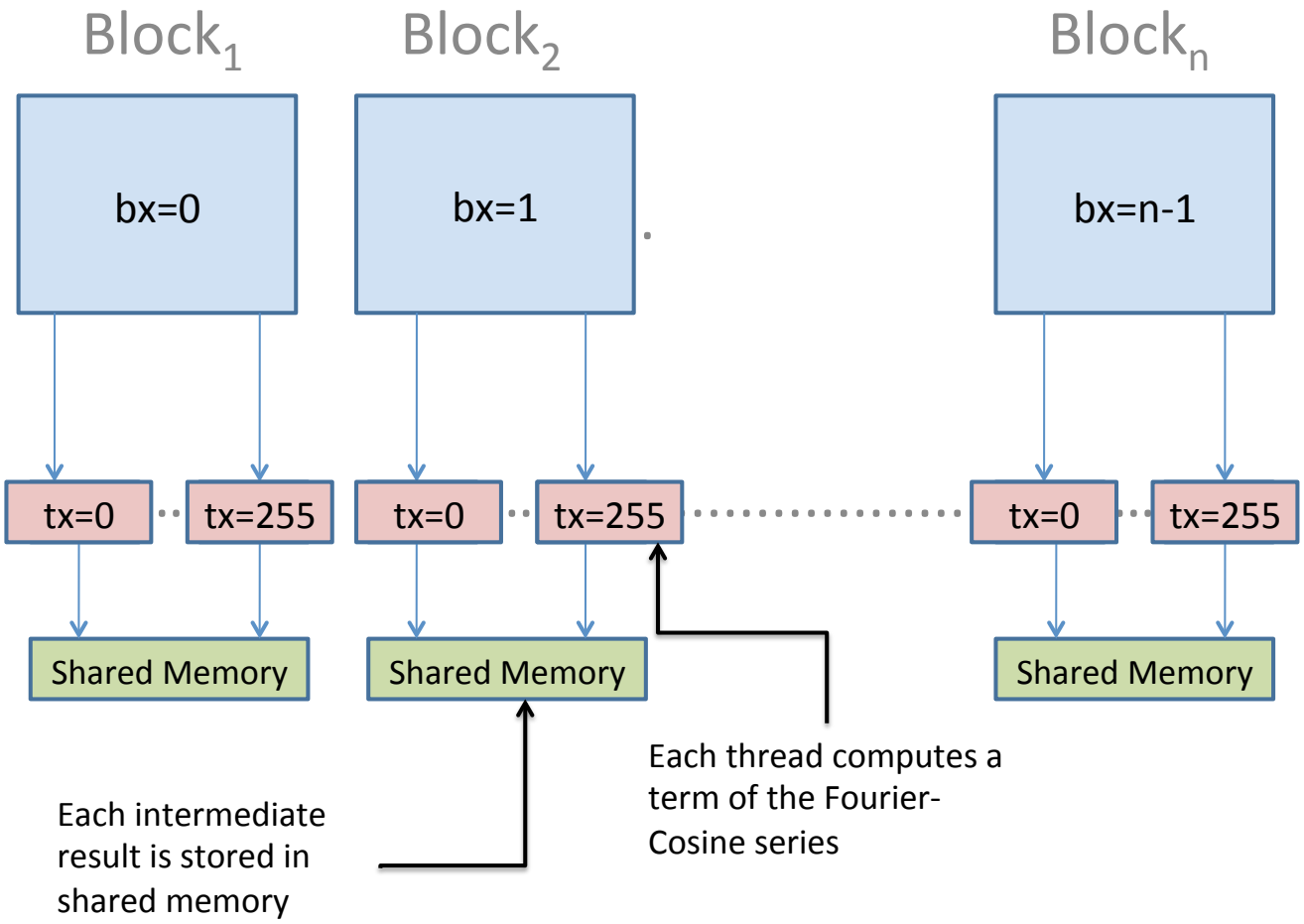
Table : Performance results for the C code in seconds.

GPU Implementation



GPU Implementation

Each option in the chain is mapped to a thread block

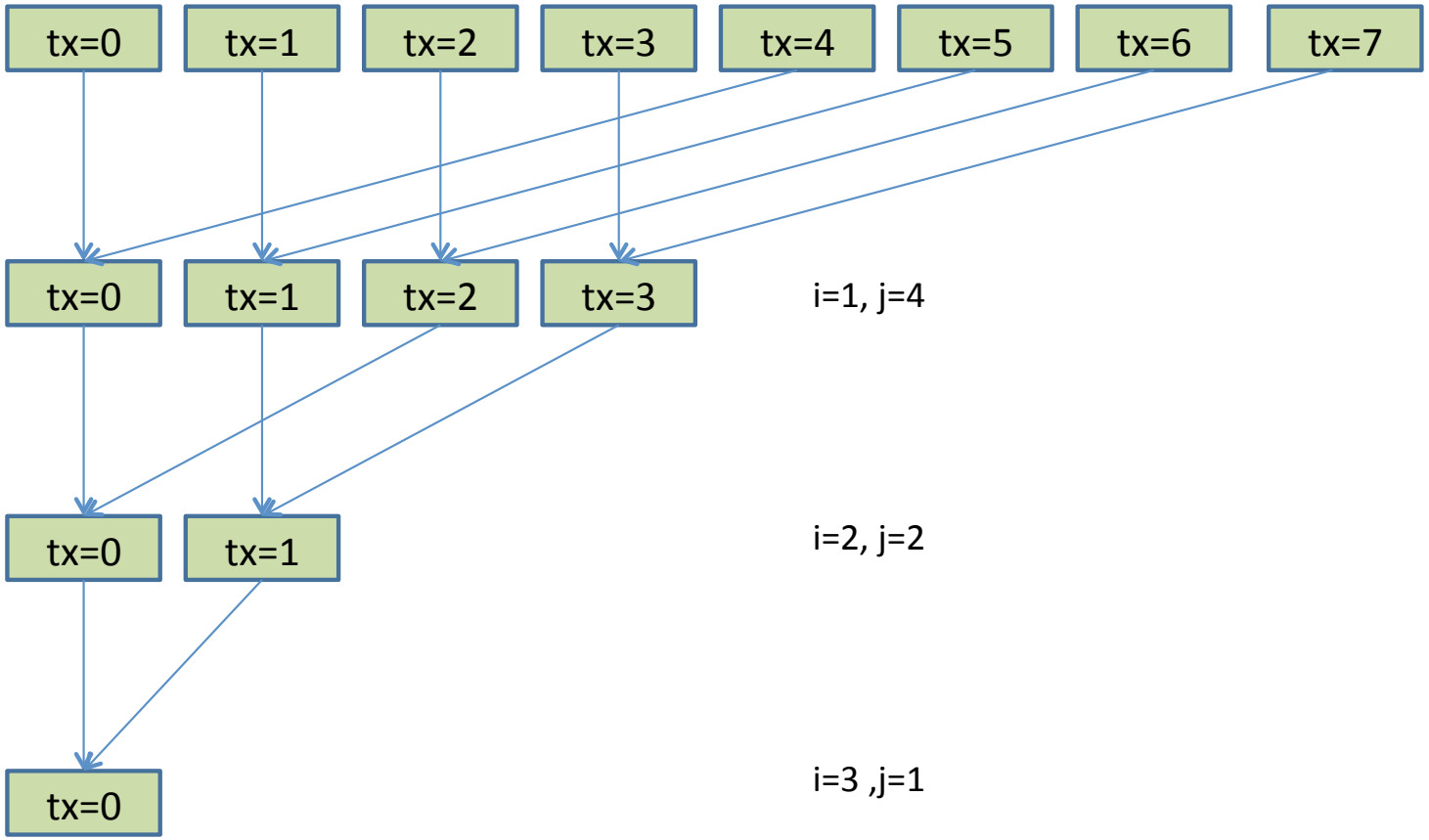


Parallel-Fourier-Cosine(p)

```
shared memory smem[]
tx ← threadIdx.x
bx ← blockIdx.x
bd ← blockDim.x
j ← bd
smem[tx] ← CHARACTERISTICFUNCTION(T[bx], p)
for i = 1 to log2(bd) do
    j ← j/2
    if tx < j then
        smem[tx] ← smem[tx] + smem[tx + j]
    end if
end for
if tx = 0 then
    V[bx] ← K[bx] × exp(−r0 × T[bx] × smem[0])
end if

return V[bx]
```

Parallel reduction using shared memory



Overview of the *gpusvcalibration* library

- The `gpusvcalibration` accelerates stochastic volatility model calibration by off-loading the error function on to the GPU
- The package is designed to hide the parallelism from the R user
- The package is designed for use with existing non-convex optimization CRAN packages such as `DEoptim` and `nloptr`
- The library currently supports European option pricing under four different stochastic volatility models and more models are planned for the future

R Wrapper for C++/CUDA implementation

```
ErrorFunction<-function(z){
       if (!is.loaded('gpuMapReduce')) {
       dyn.load('gpuMapReduce.so')
    }
   RMSE<-.Call("ErrorFunction", as.numeric(p))
   return (RMSE)
}
```

Sample code for performance benchmarking *gpusvcalibration*

```
library("gpusvcalibration")
library("DEoptim")
library("nloptr")
chain <- Load_Chain(fileName)
Copy_Data(chain)
Set_Model('Heston') # {"Heston","Bates","VG",CGMY"}
Set_Block_Size(256)

l     <- c(eps,eps,eps,-1.0 + eps, eps)
u     <- c(5.0-eps,1.0-eps,1.0-eps,1.0-eps,1.0-eps)
args  <- list(NP=100, itermax=25)
DEres<- DEoptim(fn=Error_Function, lower=l, upper=u, control=args)

res   <- nloptr(x0=as.numeric(DEres$optim$bestmem), eval_f=Error_Function,
            lb = l, ub = u,
            opts=list("algorithm"="NLOPT_LN_COBYLA", "xtol_rel" = xtol))
Dealloc_Data()
```

Performance benchmarks of the GPU implementation[3]

	AAPL	AMZN	BP	CSCO	GOOG	MSFT
DEoptim	0.31	0.12	0.08	0.074	0.29	0.08
nloptr	0.16	0.063	0.044	0.041	0.15	0.044
ErrorFunction (ms)	1.41	0.56	0.36	0.33	1.25	0.36
Total ErrorFunction	0.42	0.17	0.11	0.1	0.38	0.11
Total Time	0.53	0.23	0.17	0.16	0.44	0.18
% ErrorFunction	79.2%	73.9%	64.7%	62.5%	86.3%	61.1%

Table : Performance results for the RGPU code. Timings are shown in seconds unless stated otherwise.

[3] Based on an Intel Core i5 processor and NVIDIA Tesla K20c (Kepler) with 2496 cores

Sequential implementation of the calibration program in R

	AAPL	AMZN	BP	CSCO	GOOG	MSFT
DEoptim	293	111	69	62	255	70
nloptr	148	56	35	32	130	36
ErrorFunction	1.46	0.55	0.34	0.31	1.28	0.35
Total ErrorFunction	440	166	103	93	385	105
Total Time	441	167	104	94	386	106
% ErrorFunction	99.8%	99.4%	99.1%	99.0%	99.7%	99.1%

Table : Performance results for the R code in seconds. Each columns represents a different option chain.

Comparative Performance of the GPU implementation[4]

	AAPL	AMZN	BP	CSCO	GOOG	MSFT
R/RGPU	1042	992	971	933	1024	961
C/RGPU	313	297	288	278	303	297
CGPU/RGPU	1	1	1	1	1	1

Table : Performance comparison of the offloaded *ErrorFunction* to a serial *R* implementation, a serial $C/C++$ implementation and a *CGPU* implementation.

[4] Based on an Intel Core i5 processor and NVIDIA Tesla K20c (Kepler) with 2496 cores

Summary

- Stochastic volatility (SV) models are used extensively across the capital markets for pricing and risk management of exchange traded financial options.
- Calibration of SV models is computationally intensive and requires specialist optimization routines
- Our *gpusvcalibration* R package accelerates SV model calibration on GPUs and can be used with R optimization packages.
- Demonstrated a factor of up to 760x performance improvement on a NVIDIA Tesla K20c (Kepler architecture)

Resources

- Paper:
  M.F. Dixon, S. Khan and M. Zubair, gpusvcalibration: A R Package for Fast Stochastic Volatility Model Calibration using GPUs. http://ssrn.com/abstract=2394306.
- Development version of R package:
  https://github.com/mfrdixon/gpusvcalibration