Accelerated Root Finding for Computational Finance

Mark J. Bennett
Rho Trading Securities, LLC
Chicago, IL, USA
markb@rhotrading.com

Abstract—**A parallel implementation of root finding on an SIMD application accelerator is reported. These are roots of stochastic differential equations in the computational finance domain which require a stochastic simulation to be performed for each evaluation of the pricing function. Experiments show that a speedup of 15X can be achieved over using a stand-alone CPU processor, depending on the required accuracy, and the numerical method employed.**

Keywords-high performance computing; SIMD computer architecture; numerical methods; valuation; financial derivatives

I. INTRODUCTION

Traditionally, computational finance used multiple instruction multiple data stream (MIMD) such as Blade computers [1, 2], but that is changing with the availability of special application accelerators. Many problems in high performance computing for finance, such as derivative valuation and risk measurement, are intensively multi-dimensional and floating point in nature to the point where parallel algorithms can be oriented toward single instruction multiple data stream (SIMD) coprocessors. An implementation with a single coprocessor is discussed.

II. HARDWARE

One or more SIMD coprocessors are added on the system bus. The CPUs and coprocessors communicate over the system bus. While several application accelerators can be added to the system bus, the results reported here are for 1 CPU core with and without 1 SIMD coprocessor.

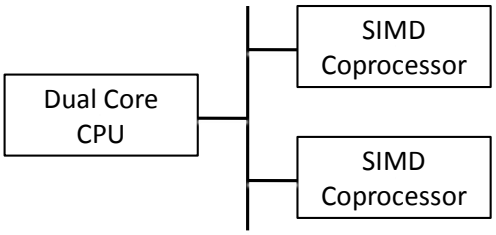

Figure 1. Hardware Architecture.

III. ALGORITHM IMPLEMENTATION PHASES

The computational finance application domain hierarchy is now described. An investor or firm has a *portfolio*, which is a set of *positions* held in one or more financial *instruments*. Each of those instruments usually has one or sometimes many underlying *securities*. Many thousands of potential market *scenarios* need to be simulated over *future market time* t for each security. The overall cardinality of the simulation problem becomes larger when the investor or firm is larger. An illustration of these three phases for a given position valuation under just two market scenarios will be presented. Many times the underlying securities are stock prices which are modeled as continuous random variables which vary over time $S(t)$ where $t \geq 0$ and $S(t) \geq 0$.

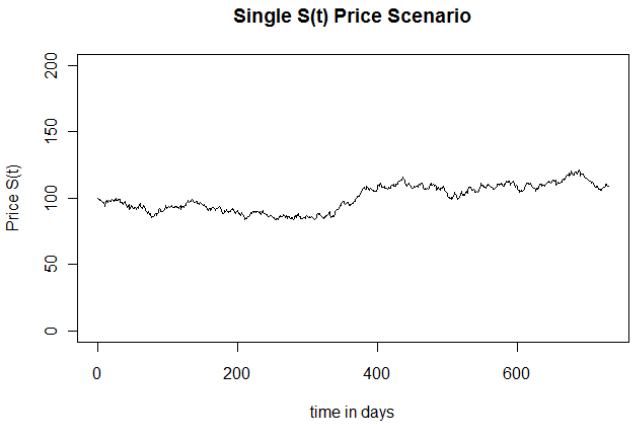

Figure 2. One Scenario of Stock Price Evolution.

In quantitative financial applications, algorithms have 3 distinct phases, scenario generation, valuation and root finding, which are explained below. The first two phases will be also known as pricing.

A. Scenario Generation

Scenario generation is a simulation of the market prices moving according to an agreed upon and customary probability density function. This is a solution to a set of stochastic differential equations (SDE) which has as many dimensions as the number of underlying securities in the entire portfolio. Thousands of scenarios need to be simulated in order to account for potential market situations. Geometric Brownian Motion (GBM) models the behavior of stocks price

movements over simulation time, $S(t)$ in terms of the drift and volatility, μ, σ with stochastic term $dW(t)$.

With GBM it is assumed that stock prices moves randomly over time according to the lognormal distribution. With the lognormal distribution, the natural logarithm of the rate of return of the stock prices at a future time T as compared to now, t, $S(T)/S(t)$ is normally distributed. In this case, T (in decimal years) is the maturity of the instrument. For example, three months is represented as $T = .25$.

$$\ln(S(T)) - \ln(S(t)) \sim \phi[(\mu - \sigma^2/2)(T-t), \sigma\sqrt{T-t}] \quad (1)$$

where $\phi[m, s]$ is the normal distribution with mean m and standard deviation s.

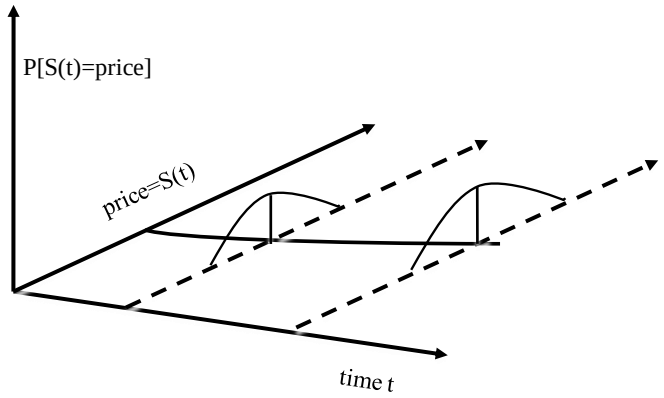

Figure 3. Probability Distribution of Stock Prices over time for a Single Process of Equations (2).

Geometric Brownian Motion is the assumption of the Black-Scholes formula, an industry standard. Each derivative position relies on one or more underlying securities; in this case stock price variation over time is denoted by $S(t)$.

$$dS_1(t) = \mu_1 S_1(t)dt + \sigma_1 S_1(t)dW_1(t) \quad (2)$$

…

$$dS_m(t) = \mu_m S_m(t)dt + \sigma_m S_m(t)dW_m(t)$$

There is one of these equations for each underlying security in the portfolio [4][5][6][7].

B. *Valuation*

A portfolio of derivative positions is valued and this requires millions of simulation steps over the time horizon, applying a known payout function which models the financial instrument applied of the market prices using a set of input parameters. An instrument which is priced on one underlying security has only one random process to consider, but some exotic instruments can take into account as many as 30 underlying securities. In any case, the valuation forms a boundary value problem for the SDEs.

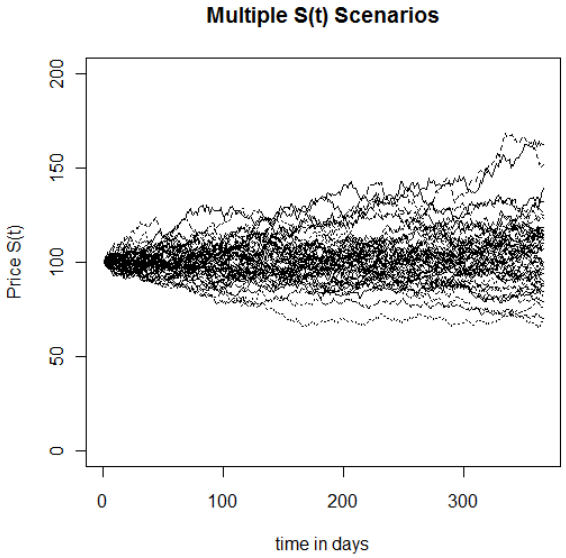

Figure 4. 100 Scenarios of Stock Price Evolution $\mu = .05, \sigma = .20$

Scenario generation and valuation involves evaluating v in

$$v(x_i, \mathbf{a}_i, t) = y_i \quad (4)$$

where $v(x_i, \mathbf{a}_i, t)$ is a formula involving the expected value of parameters in the vector $\mathbf{a}_i$.

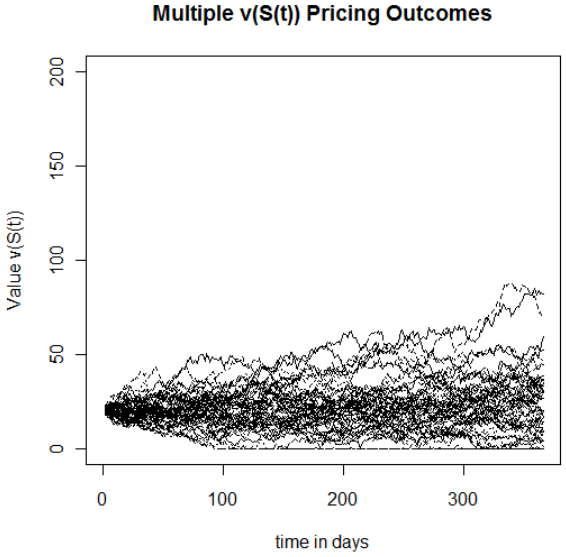

Figure 5. Example of 100 Scenarios for Pricing $\mu = .05, \sigma = .20$

C. *Root Finding*

Root finding involves applying a numerical procedure to finding an unknown input parameter to the valuation process. While pricing algorithms have a fixed number of iterations

until completion, root finding algorithms have a variable number of iterations until convergence.

IV. Root Finding Methodology

Root finding is equivalent to the process of finding the inverse of the pricing function at a specific point. Root finding gets its name from finding the values x of polynomial functions $f(x)$ where $f(x) = y = 0$, however, in this case, the functions are not polynomials, but are instead the x-values or the inverse of pricing function v known as v^{-1}. They are the roots of the stochastic differential equations. More specifically, if Equation (4) holds then $v^{-1}(y) = x$ and the inverse function v^{-1} can be expressed as a root finding problem in Equation (5). Given y-values, successively more accurate x-values are refined until Equation (5) is approximately satisfied. Newton-Raphson and Bisection [8] are common techniques to generate successive x-values for convergence.

$$v^{-1}(y_i, \mathbf{a}_i, t) - x_i = 0 \tag{5}$$

A. Vectorized Root Finding Algorithm

The vector pricing function and its inverse function which the root finder is intending to find is listed below.

$$\mathbf{v}(\mathbf{x}, \mathbf{A}, t) = \mathbf{y}, \mathbf{v}^{-1}(\mathbf{y}, \mathbf{A}, t) = \mathbf{x}, \tag{6}$$

Pricing, the evaluation of $\mathbf{v}$, occurs in a massively parallel way on the coprocessor. This is where the bulk of parallelism is achieved both within computing the pricing function $\mathbf{v}$ and the parallelism between different positions each invoking $\mathbf{v}^{-1}$.

Typically, root finding convergence speed determines the amount of iteration performed. When vectorizing the root finder, a static rather than dynamic iteration limit is used. This is done in order to feed to the coprocessor the maximum number of calculations at each iteration step. All conditional logic is performed in parallel on each of the desired solutions.The vectorized root finding primitives are invoked on the SIMD coprocessor as depicted in Figure 6. On a CPU the variation in iterations is of small consequence in a large portfolio. Iteration ceases upon convergence for each position and the CPU moves on the next position without wasting cycles. With the coprocessor, iteration involves a series of kernel invocations. While some elements of the vector have achieved convergence, there are other elements that have not. As iteration continues, thread cycles are wasted on those vector elements which have already converged. After implementing several root finding algorithms, the conclusion is that vectoring root finding achieves the best speedup when the deviation of convergence iterations is smallest.

B. Accuacy Check

The parallel root solution for v^{-1} is called $\bar{\bar{v}}^{-1}$. It and the serial root solution v^{-1} are compared for the M positions of the portfolio and compared to the aggregate tolerance as in Equation (7).

$$\sum_{i=1}^{M} \left| \bar{\bar{v}}^{-1}(x_i, \mathbf{a}_i, t) - v^{-1}(x_i, \mathbf{a}_i, t) \right| < \varepsilon \tag{7}$$

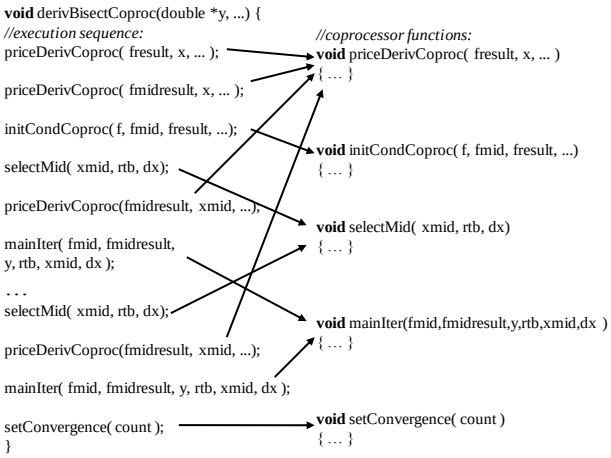

Figure 6. Invocation of Vectorized Coprocessor Functions

C. Speedup

Speedup of 15X or greater between an unaccelerated CPU and a CPU accelerated with a single SIMD coprocessor can be achieved as shown in the experiment results of Figure 7 for portfolio size M.

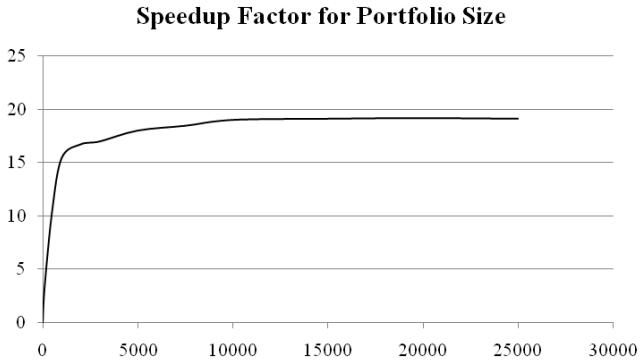

Figure 7. Speedup from Experiment with Single Coprocessor

References

[1] Wright, J., "Blades Have the Edge," Spectrum, IEEE, Volume 42, Issue 4, April 2005 pp. 24 – 29

[2] "Algo Suite Technical White Paper," Algorithmics, Inc., Hewlett-Packard, April 2005.

[3] F. Black, M. Scholes, "The Pricing of Options and Corporate Liabilities," Journal of Political Economy, 81 (May-June 1973), pp. 637-659.

[4] J. C. Hull, *Options, Futures and Other Derivatives*, Prentice Hall, 2006.

[5] G. Levy, *Computational Finance using C and C#,* Academic Press, 2008. G. Levy, *Computational Finance using C and C#,* Academic Press, 2008.

[6] S.E. Shreve, *Stochastic Calculus for Finance: Volumes I and II*, Springer Finance, 2004.

[7] M. J. Bennett, "Computational Finance," Presentation to Chicago Chapter of the Association for Computing Machinery, December, 2008.

[8] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C*: *The Art of Scientific Computing*, Cambridge University Press, 2007.