J Supercomput (2009) 47: 146–170
DOI 10.1007/s11227-008-0182-4

A software architecture framework for on-line option pricing

Kiran Kola · Ruppa K. Thulasiram · Parimala Thulasiraman

Published online: 21 March 2008

Abstract The problem of growing computational complexity in the finance industry demands manageable, high-speed and real-time solutions in solving complex mathematical problems such as option pricing. In current option trading scenarios, determining a fair price for options "any time" and "anywhere" has become vital yet difficult computational problem. In this study, we have designed, implemented, and deployed an architecture for pricing options on-line using a hand-held device that is J2ME-based Mobile computing-enabled and is assisted by web mining tools. In our architecture, the client is a MIDP user interface, and the back end servlet runs on a standalone server bound to a known port address. In addition, the server uses table-mining techniques to mine real-time data from reliable web sources upon the mobile trader's directive. The server performs all computations required for pricing options since mobile devices have limited battery power, low bandwidth, and low memory. We have parallelized and implemented various computational techniques such as binomial lattice and finite differencing. To the best of our knowledge, this is one of the first studies that facilitate the mobile-enabled-trader to compute the price of an option in ubiquitous fashion. This architecture aims at providing the trader with various computational techniques to avail (to provide results from approximate to accurate results) while on-the-go and to make important and effective trading decisions using the results that will ensure higher returns on investments in options.

Keywords Mobile/ubiquitous computing · Web table-mining · Finance applications · Option pricing algorithms · MIDP · J2ME

K. Kola · R.K. Thulasiram (✉) · P. Thulasiraman
Department of Computer Science, The University of Manitoba, Winnipeg, MB, Canada
e-mail: tulsi@cs.umanitoba.ca

K. Kola
e-mail: kirankkk@cs.umanitoba.ca

P. Thulasiraman
e-mail: thulasir@cs.umanitoba.ca

1 Introduction

This paper is a study of the use of mobile technology combined with internet mining for successful trading that would pave way for effective investment trading towards better profitability through higher returns. In effect, this study combines e-commerce and e-finance with the modern mobile technology to create M-commerce environment. Option pricing forms a fundamental objective and backbone of financial risk management and decision-making solutions in option trading. Active trading takes place either at the trading floor or through computers with instructions from investors or investment managers. However, once an investor steps away from the workplace, the investor encounters problems of interrupted trading, as the required information is no longer available. In such cases, investors have to rely on the data provided by some other sources (such as electronic board display). If the investor is away from the building, he/she has to be in continuous touch with some other sources such as a broker, to get some basic information about the market to analyze the market behavior. However, the information provided by intermediary brokerage firms is generally inadequate especially in the case of computing the option values.

Futures and *Options* are the most common type of derivatives and both are actively traded on many exchanges. Our current research focuses only on the option pricing and therefore, the discussion of futures and other securities is beyond the scope of this paper.

In pricing options, every investor must understand the possible future trends of an underlying asset and it is potential for speculation and hedging. In the derivative market, accuracy and ability to respond ubiquitously to the fluctuating market is extremely important for every active investor. Therefore, to achieve ubiquitous nature in option trading we need an infrastructure, which will enable the trader to avail various computational techniques for accurate and immediate results. This can accelerate decision-making process. Mobile technology is a new technology that rides a new wave of business innovation. Use of mobile technology for e-business and decision-making strategy is slowly changing the dialogue between investors and traders on the floor of a stock exchange into M-business deals.

In this study, we focus on three major issues to achieve ubiquity in derivative markets: (i) mobile commerce aspects in derivative markets (particularly financial options), (ii) various computational techniques used to price options, (iii) mining a real-time finance data from web sources. We have incorporated all these issues to provide a value-added, ubiquitous service to the trader on the go. We use the terms ubiquitous, and pervasive interchangeably in this paper.

In our preliminary work [23], we have done a feasibility study of derivative pricing using a short-range wireless connectivity with a PDA. Our goal in the current study is to enable wireless trading strategies in ubiquitous fashion and ensure portability for the trader. In the current work, we are experimenting and validating our architecture on J2ME-based mobile emulators, which is applicable for limited and broad range of wireless range networks. To make our architecture more effective, we use parallel computing to do the computations at hand.

The rest of the paper is organized as follows: In the next section, we introduce some vocabulary relevant to the current study with a motivation for Mobile-Pricing

(M-pricing) in option trading. We have presented some background details on mobility in derivatives market and option pricing in Sects. 3 and 4, respectively. We then describe our overall architecture design for real-time option pricing in Sects. 5 and 6. We discuss a set of results in Sect. 7 and our conclusions in Sect. 8.

Our contribution through this work is the design and development of an architecture that enables mobiquitous pricing using high performance parallel computing in the emerging business scenario.

2 Vocabulary in option trading

An option [1] is a security that gives its owner the right without creating any obligation to trade-in a fixed number of shares of a specified asset (e.g., stocks) at a fixed price (strike price) at any time on or before a specified future (maturity) date.

2.1 Fundamental ideas

There are two parties involved in the option trading namely holder and writer. The holder of the option gets the right to buy (call) or sell (put) assets at a predetermined time in the future for a predetermined value; the writer of the option is obliged to deliver (call) or take delivery (put) of the underlying asset.

Two basic styles of options are *European* and *American*. While the former can be exercised only at maturity, the latter can be exercised at any time prior to maturity. Every option has a set of parameters that are required to compute the price of the option. These are strike price, stock price, risk free interest rate, period of contract, and volatility of the underlying asset. The *strike price* of a call (put) option is the contractual price at which the underlying asset will be purchased (sold) in the event that the option is exercised. The risk-free interest rate (r) is the rate at which an investment (such as simple deposit) would grow without incurring any risk to the capital. The time in years until the expiration of the option is called *maturity date*. A measure of the change (either up or down movement) of the underlying security over a given period is known as *volatility* (σ).

In option markets, accuracy and ability to respond quickly to the fluctuating market is vital for every active investor.

2.2 Motivation for ubiquitous pricing

The use of mobile technology extends the nature and scope of the e-commerce providing additional advantages by enabling continuous communication and information access regardless of locale, thus resulting in continuous touch with the market. Ubiquity in the market place is concerned with the use, applications, and integration of diverse services. Many researchers attempted to address innovative research issues and possible solutions in mobile commerce. Research efforts by Varshney and Vetter [2] emphasize that mobile financial application is one of the important component of M-commerce, which can replace banks, ATMs, and manual methods by wireless aided services such as on-line brokerage, and micro payments, etc. Readers can find several interesting issues and research problems that are related to mobile commerce in [3–6]. The current research is motivated by the concept of providing value-added services in option trading to trader-on-the-move and is driven by the following principles [7, 8].

Operational focus Traders have access to handheld devices (mobile device or any wireless device) all the time. Therefore, as a value-added service, a trader should be able to compute the option price irrespective of his/her physical location. This type of trading is called Martini trading (trading any time and anywhere). A recent report on Mobile Commerce (M-Commerce) by Durlacher Research states "The ability to receive information and perform transactions from virtually any location is especially important to time-critical applications, such as stock and options trading as well as betting. Providing mobile traders with a similar level of access and information to that available in the fixed line environment is the key."

Personalization Wireless devices like mobile devices are typically operated by and configured for a single trader. Thus, the trader can receive personalized information on his/her investments as and when new information (for example, new price, new contract, etc.) becomes available.

Multi-channel trading Customer prefers a choice in the channel through which they do business. When a trader cannot associate himself with the main trading terminal as in cases such as (i) having power failure (ii) being away from the trading floor and (iii) experiencing a nonfunctional terminal, a mobile device can act as an alternative trading terminal.

Handiness Mobile devices are compact, low-cost, and have improved security, thus, making them popular for wireless trading.

3 Mobile computing developments related to trading

Mobile trading is a logical extension of e-business to address new customer channels and opportunities that deal with many services and applications in ubiquitous fashion. Even though little is reported on the confluence of derivative trading with wireless devices, researchers have recognized the need for such incorporation. In this section, after listing possible benefits of ubiquity in trading, we classify the use of the ubiquity into three broad aspects (i) commodity trading (ii) risk management and (iii) services and software. The frequent fluctuations in prices or the increasing volume of transactions are often overwhelming due to dependence on time-critical information of breaking news on a company's earnings, losses, or a change in their structure. In order to make conscious decisions in an uncertain market place, every investor needs time-critical information in a ubiquitous fashion [7].

Kargupta et al. [9] justify the needs and benefits of reporting time-critical information of stock data through wireless networks. These authors have introduced a mobile data mining system that facilitates intelligent monitoring of time critical financial data from a hand held device such as PDA (Personal Digital Assistant) and mobile.

Ajenstat [10] proposed a new idea for automation of on-line derivative (stock options) trading in any place and at any time without the presence of a decision maker. More specifically, a *virtual decision maker* is a network of cooperating intelligent agents. These agents focus on some rule-based environment and expert-validated knowledge to analyze the behavior of the stock market. Furthermore, the author emphasizes that the integration of wireless commerce (option of accessibility) with web-based virtual decision-making systems will play a vital role in the future.

For our research, we integrate time-critical information pervasively. That is, with predefined threshold and boundaries on the price movements of the asset in question, our architecture will initiate new computation in a pervasive fashion whenever "real-time" price of the asset deviates from a predefined value or a predefined range. The "real time" prices on an asset are monitored continuously through collection of this information from on-line web sources with wireless devices. For instance, time-critical information for option pricing such as "volatility," and "prime rate" are mined from reliable web sources (presently Yahoo! Finance and Money Cafe). In addition, we have developed our architecture to choose one computational technique (at a time) among various techniques that we have implemented on our back end server. These techniques exploit the time-critical information in order to compute the price of an option accurately for a particular underlying stock. In the following section, we describe some of the works related to the requirements of mobility in the derivative market.

3.1 Round-the-clock trading

There are several factors behind the large dependence on wireless trading. Couples of most compelling factors are: first, trading activity is becoming a 24-hour-a-day business—especially for those who invest in commodities have to be following the Asian and European markets, in sync with the North American market; there is higher potential for a trader to miss market movements without wireless connectivity. In other words, wireless devices make this possible, which enable the trader to track the various market movements anytime or anywhere; second, Roche [11] states that, on black Monday (1987), one option trader lost £55,000 in 5 minutes just for leaving the market for such a short period. Research by Roche [11–13] focuses on the need for wireless services in equities and derivative markets. The author emphasizes that several market participants would prefer on-line trading while on the move, due to many advantages: (1) Traders and brokers who are not bound to desktop trading such as farmers or exporters; (2) General investors who want access to the market information while on the go; (3) Traders and brokers who need assurance of an alternative-trading terminal in case their main terminal gets disconnected.

Trading derivatives with hedging strategies has become one of the most important recent developments in the financial market. Currently, investors price options on-line from their desktop computers using various software tools [14–16]. However, once an investor steps away from the work place, he/she is disconnected from the market place. There have been some recent advancements to aid traders on the move, SMS (Short Message Service) being one of them. "Push" and "pull" models are specific M-commerce services in wireless trading. In the push model, text messages are sent to a wireless device via SMS without any previous trader request. Examples of push models include text messages sent to a mobile phone to alert clients regarding financial news. The expansion of SMS-related services reflects higher volumes of remote stock trading. Presently, a trader can subscribe to services such as CommSec [17] and Quo Trek [18] in order to be connected on the move. Technically, by subscribing to *equity-alert-services*, one can receive personalized equity information (real-time price information for personalized stock options) over a mobile phone. On the other hand, in pull messaging, mobile client invokes server-side applications and the resulting output is pulled from the web-server. However, information provided by brokerage firms, via SMS, is insufficient (for only limited information on the asset and period of contract is available through this message). Thus, the information from brokerage firms cannot be used to compute the option price for a particular option contract.

As a first step toward an organized use of mobility devices, Mobility Partner Advisory Council recently announced [19] that "The Chicago Board of Trade (CBOT) is deploying up to 10,000 wireless enabled pocket PC devices in the two years (2004–2005) to floor traders to automate the trading process."[1]

The research effort by Web [20] focuses on software solutions for hedge fund-managers. According to Web, investors are becoming far more cautious and, in fact, want to see the risk management and reporting system before they invest on any asset. Moreover, they are not ready to accept basic risk analysis offered by hedge fund administrators. Roche [12] focuses on the risk management needs in the area of farm production and financing. The author emphasizes that incorporating commodity risk management into global on-line trading will significantly increase the use of commodity risk management tools worldwide. In addition, the integration of wireless communication and the commodity market, through appropriate software and hardware, will increase the agriculture derivative's business with consequent benefits.

Wireless traders do not have the time or power to browse on-line information from hand-held devices and calculate the risk level of a particular asset. However, a trader seeks personalized information to be delivered in a ubiquitous fashion. Chang and Cheng [21] emphasize a need of wireless solution for derivative trading and risk management. An international task force on commodity risk management explored new mobile assisted market-based approaches, in order to manage their vulnerability to commodity price fluctuations. Furthermore, Chang and Cheng [21] stress that such market-based risk management strategies increase the trading volumes of commodity derivative.

Patsystems [19] has developed software called H-trader, which assists a trader to do Martini Trading. H-trader operates on a mobile phone to trade derivatives across the world. According to Patsystems, the integration of an H-trader with some risk management tools will make more efficient trading environment. Thinkorswim [22], a brokerage company in Chicago, enables the registered trader to perform trading (stock options, and other securities) via web-based, PDA, or Mobile devices. Using

[1]CBOT provides wireless technology for commodity and option trading through Master Antenna System. For secure trading, other devices such as Bluetooth enabled hand held devices are prohibited from the trading floor.

these services, the trader can follow real-time audio commentary in stock exchanges and simultaneously has a feature of selling or buying commodities. Windale technologies [15] and FIS-Group [16] present innovative pricing software with various option-pricing techniques that evaluate American and European style call and put options. In addition, support for both desktop environment and Pocket PC versions is available.

The ineffectiveness of these technologies is as follows: (a) The above-mentioned enterprise versions are high-priced products; (b) Frequent changes in these products require new updates to the software each time; (c) The computing technique(s) employed for option pricing and their working principles are not explained in their products to the end user; and (d) These products do not present the trader with real-time prices and other parameters (prime interest rate, volatility). Unfortunately, parameters (for example, volatility) used in the computational techniques are highly sensitive to the market fluctuations.

In the current study, we have used various computational techniques such as binomial lattice, finite differencing and fast Fourier transform, etc. For each technique, our goal is to use real-time information that is mined from reliable web-sources such as Yahoo! Finance. Our architecture uses the cost-effective implementation of a client and server scheme (for instance, Apache Server, and MIDP-interface development are open source environments) [29]. In addition, if there are any new updates in computation techniques, we just need to upload the techniques to the back end server rather than uploading them to the client. This way, the client saves the overhead cost each time when there is a new update. Due to lack of space, we do not describe the algorithm for web mining related to option pricing. We refer the reader to our earlier work [23]. In what follows, the design and implementation explanation to make mobile trading a possibility using the current technology, we propose architecture to address this problem.

4 Computational techniques for option pricing

Many techniques, such as the binomial lattice method, the finite differencing method, and the Monte Carlo method are being used for pricing options. We describe below two computational techniques for pricing options: (i) a recent advance known as finite-difference technique to solve financial models manifested as partial differential equations and (ii) a classical binomial lattice technique. We have parallelized both the algorithms and details of the parallel algorithm are described for one of the algorithms (binomial lattice). We have implemented other computational techniques and for lack of space, we do not describe them here. More information on these and other methods can be obtained from [24–27, 37]. We can use any of these techniques as stand-alone modules for option pricing in our architecture described in Sect. 6 and we can incorporate any of the latest pricing techniques in addition to these modules, if necessary.

4.1 Finite differencing technique

Finite differencing technique is a fundamental numerical approach for pricing financial securities. There are several finite-difference schemes available such as, Mc-

Cormack scheme, Richardson scheme [36]. Consider the Black–Scholes model [40], a classical option-pricing model

$$\frac{\partial u}{\partial t} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 u}{\partial S^2} + rS\frac{\partial u}{\partial S} - ru = 0 \tag{1}$$

where u is the option price, t is time, σ is volatility, S is the asset price, and r is the interest rate with initial and boundary conditions: $u(0;t) = 0$; and $\lim_{S\to\infty} u(S;t) = S$, $u(S;T) = \max(S-E;0)$ for a call option and $u(S;T) = \max(E-S;0)$ for a put option. We will only consider a call option here. The appropriate discretization for each term in (1) is dictated by the individual terms of the PDE together with the required precision and performance constraints. The accuracy of the results can be controlled by the use of a finer grid in the computational time direction as well as the space direction. That is, we iterate the solution process over many computational time steps until we reach a steady state solution. For our research, we have implemented parallel FTCS finite-difference scheme to price options. This is a forward differencing in time direction and central differencing in space direction (for further details on finite-difference schemes, please refer to [33]).

We transform the B-S model into a form of a diffusion equation that makes it easier to apply the FTCS finite-difference method to price options. Assuming $S = Ee^x$, $t = T - \tau/(\sigma^2/2)$ and $u(S,t) = g(x,\tau)v(x,\tau)$ where E is the strike price, $g(x,\tau) = Ee^{-\frac{1}{2}(k_1-1)x-\frac{1}{4}(k_1+1)^2\tau}$, in which $k_1 = 2r/\sigma^2$ the Black–Scholes equation reduces [35] to $\frac{\partial v}{\partial \tau} = \frac{\partial^2 v}{\partial x^2}$ subject to initial and boundary conditions (for the European call option)

$$v(x,0) = \max\left(e^{-\frac{1}{2}(k_1-1)x-\frac{1}{2}(k_1+1)\tau}, 0\right)$$

$$\lim_{x\to-\infty} v(x,\tau) = e^{-\frac{1}{2}(k_1-1)x-\frac{1}{4}(k_1+1)^2\tau}, \qquad \lim_{x\to\infty} v(x,\tau) = 0$$

To discretize the above equation, a finite number of equally spaced time steps between the current date ($t = 0$) and the maturity date of the option, ($t = T$) are chosen. Similarly, a finite number of equally spaced asset prices (Nj) are also chosen. $\Delta t = T/N$, $(N+1)$ total time steps and, $\Delta S = S_{\max}/2Nj$, $(2Nj+1)$ total asset prices. By the above discretization, a grid consisting of a total of $(N+1)(2Nj+1)$ points is constructed as shown in Fig. 1. The grid point (i,j) corresponds to time $i\Delta t$ and price $j\Delta S$. The third dimension is the computational time step (note that this time step is different from physical time step, which marches from expiration to the current date) which is used to ensure the stability of the results.

The solution scheme is iterated over many *computational* time steps until it reaches a steady state. Steady state is defined as a scenario when the solution changes very little between two consecutive computational time steps. Figure 1 shows an instance during the computation of the option price. The computed values of the horizontal layer $(c-1)$th are used to calculate the values of layer c. The number of computational layers depends on the relative error. To calculate the option values at the cth layer at grid point (i,j), we use the values from the $(c-1)$th layer, which can be expressed as

$$u^c_{i,j} = \rho u^{c-1}_{i+1,j-1} + (1-2\rho)u^{c-1}_{i+1} + \rho u^{c-1}_{i+1,j+1}$$

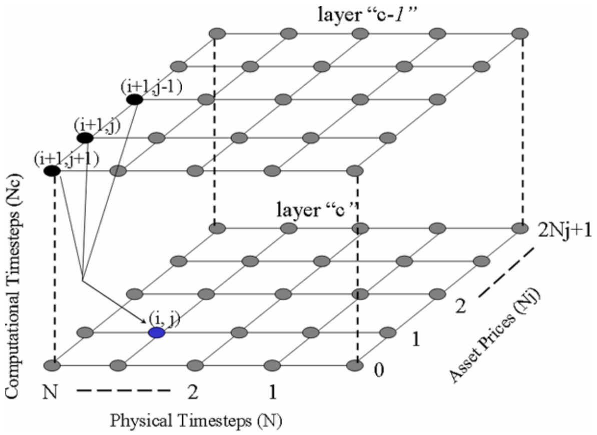


Fig. 1 Option pricing computational domain

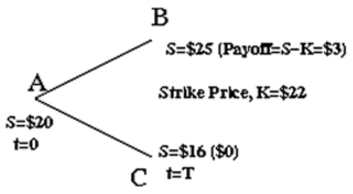


Fig. 2 One-step lattice

Here, $\rho = l/h^2$, where l denotes step size in time direction τ and h denotes step size in price direction x. If τ is from 0 to T and x is from $x_{\min}$ to $x_{\max}$, then $N \times l = T$ and $2Nj \times h = x_{\min} - x_{\max}$. Therefore, note that the terminal condition $t = T$ is transformed to the initial condition $\tau = 0$ [34]. The relative error is calculated as given below and computation stops once the "*err*" falls below certain preset threshold value: $err = u^{c}_{i,j} - u^{c-1}_{i,j}$.

4.2 Binomial lattice pricing algorithm

We describe here our pricing algorithms for options with single asset and extend them for multidimensional derivatives. These algorithms are based on the classical binomial lattice method due to Cox, Ross, and Rubinstein (CRR) [41]. This discretized approach approximates the Black–Scholes mathematical model [40] of option pricing to a large extent by representing the asset price movement in a lattice and is easy to implement for experimentation.

In the one-step binomial lattice, the tree has a root node (A) from where stock price (S) can go either up (B) or down (C) after some time T (Fig. 2). Numbers in the brackets are the possible values for a call option.

This technique can be applied to the two-step and multi-step binary trees. Two step nonrecombining tree has four leaf nodes (Fig. 3) meaning that there are four possible stock prices after some time T. Constructing the lattice like this recursively approximates the continuous-time risk-neutralized process for asset prices (it is assumed that asset prices follow a geometric process $dS = \alpha S dt + \sigma S dB_t$, where α is a constant and B_t is a standard Brownian motion).

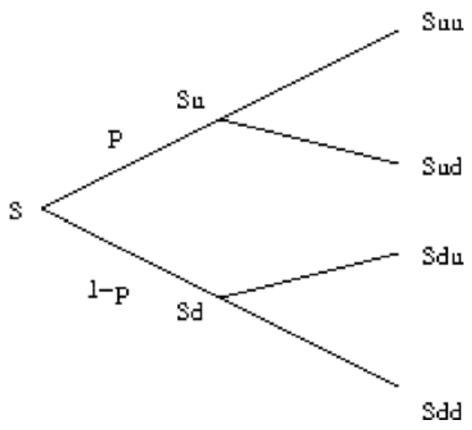


Fig. 3 Two-step non-recombining lattice

The recursive algorithm proceeds in two phases. First, we construct a lattice. Second, we work backward through the lattice from time T in order to compute the present value of the option. (Convergence of the option price computed under the binomial method to the true option price is discussed in [38].) To compute the binomial lattice, we first divide the time interval $[0, T]$ into L smaller intervals, each of length $\Delta t = T/L$ (Fig. 3). Over each subinterval, the asset price is assumed to move up from value S to Su, or down to Sd, with probabilities p and $1 - p$, respectively, as shown in Fig. 3. It can be shown that if we set $u = e^{\sigma\sqrt{\Delta t}}$, $d = e^{-\sigma\sqrt{\Delta t}}$, and $p = \frac{e^{\sqrt{\Delta t}}-d}{u-d}$ (where σ is the volatility of the underlying asset) then over each time interval, the mean and variance of asset price movements will match the mean and variance of the continuous-time risk-neutralized asset price process. In the second phase of the algorithm, we work backward from the leaf nodes at time T to compute the option price. At the expiration time T and leaf–node i, we know that the value of the option is given by $F(S_i, T) = \max[0, K - S_i]$, K is the strike price. At time $T - 1$ and node j, the value of the option is given by $F(S_j, T - 1) = \max[K - S_j]$, $(pF(S_j u, T) + (1 - p)F(S_j d, T))e^{-r\Delta t}]$, the greater of the value under immediate exercise of the option, $K - S_j$, or the expected value of holding the option for another period, $(pF(S_j u, T) + (1 - p)F(S_j d, T))e^{-r\Delta t}]$. In the recursive algorithm, communication is minimal due to evenly distributed data structure. The 1-step option pricing formula can be applied to each of the steps in computing the option value and has been extended to a multistep lattice using the above formulae. In the case of recombining tree (where the up and down movement of the prices follows certain proportions, i.e., $d = 1/u$) or in other words, $Sud = Sdu = S$, the procedure is very similar to nonrecombining tree except the fact that there are now shared values. When dealing with multistep binary trees, the techniques are exactly the same.

4.3 Parallel single asset algorithm

Let P be the number of processors. In the recursive algorithm, we delegate processor 0 as the master processor. We start at the first processor at the first level (level 0) which corresponds to time $t = 0$. We build a nonrecombining tree until the number of leaf nodes equals the number of available processors. Each leaf node is assigned to a distinct processor (one of them could be a master processor itself). Then each processor starts building its own tree. All processors are working independently on their subtrees without any information interchange. As soon as a processor creates a tree with a certain depth ($= L - \log_2 P$), it starts working backward and computes option price in every node. Each processor generates the tree and computes the option prices recursively. At the end of the computation, each processor sends its final value to the master processor. The master processor upon receiving the values corresponding to all leaf nodes from the individual processors, proceeds computing the option values recursively until the initial node at $t = 0$.

The recursive algorithm utilizes a nonrecombining binomial lattice, meaning that we do not impose the condition that $Sud = Sd = S$. The sheer number of nodes created in a nonrecombining lattice (2^L) restricts the usefulness of the recursive algorithm to short dated options. When L is large (to accurately price options with long times to expiration (large T)), the nonrecombining lattice algorithm is inefficient. We, therefore, create a recombining tree to reduce the number of nodes by utilizing $Sud = Sdu = S$. The recombining algorithm is explained next. This algorithm starts from level L.

The number of levels in the tree and the number of processors are assumed to be power of two. The number of leaf nodes is always equal to the number of levels plus one ($L + 1$). All leaf nodes are evenly distributed among the processors but the last processor receives an additional node Initially, the option prices at the leaf nodes are calculated by finding the difference between a possible stock price and strike (exercise) price, ($S_i - K$), which is the same as the local pay-off at these nodes. Every processor i, except the processor 0, send the value of its boundary node to processor ($i - 1$). (Higher numbered processor sends data to the lower numbered neighbors.) In a given processor, for every pair of adjacent nodes at a certain level L_i, the processor computes option price for pair's parent node, which is at level L_{i-1}. The computation proceeds to the previous level L_{i-1} and option price computation is repeated with additional exchange of boundary node. Eventually, the processor 0 computes the option price at the level $L = 0$.

At each time step t, both algorithms operate in two modes simultaneously: communication and computation. In the communication mode, "adjacent" processors exchange data on option values. Processor 0 only receives data from processor 1. Processors 1 through ($P - 1$) receive data from the processor's higher numbered neighbor and send data to their lower-numbered neighbor. P only sends data to processor ($P - 1$).

By forcing the lattice to recombine (imposing the condition that $Sud = Sdu = S$), we eliminated the step of constructing the lattice. The stock prices are instead calculated "on-the-fly" with proper indexing scheme. Computation at each node thus involves calculating the stock price and the option value. We calculate the asset price based on the number of up and down movements from the initial node. This calculation only requires the node and time indices. For example, the stock price at node j and time L_i is calculated as $S_j^{L_i} = S_0 * u^{L_i - 2*(L_i - j)}$.

Theoretical analysis of the recombining algorithm Here, we explain the total number of computations and communications of our algorithms. To make the explanation

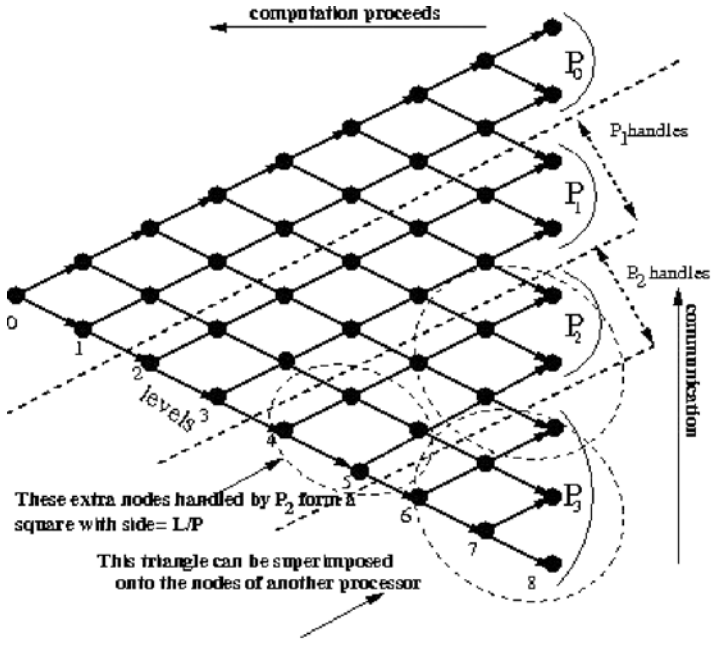

computation proceeds
P0
P1 handles
P1
P2 handles
P2
P3
communication
0
1
2
levels 3
4
5
6
7
8
These extra nodes handled by P2 form a square with side= L/P
This triangle can be superimposed onto the nodes of another processor

Fig. 4 Example with 8 levels and 4 processors with computational complexity

easier we use an example binomial tree with 8 levels and 4 processors (Fig. 4) and generalize the complexity analysis for L levels and P processors.

Total computations We are interested in counting the total number of computations performed by each processor excluding the leaf nodes. In the example, let us consider the processor P_3. The longest path from any of the three leaf nodes in P_3 ends at level 6.

The number of nodes allocated to this processor initially is $L/P + 1$ $(= 3)$. Each of these nodes has parent nodes at level 7. One of the two nodes at level 7 is boundary nodes. Note that in our algorithm, the communication is from higher numbered processors to lower numbered neighboring processor. Therefore, the boundary node is created in P_3. Similarly, these two nodes at level 7 have parent node at level 6, which is also a boundary node local to the processor. Note that all the nodes we have just discussed form a triangle starting at level 6. These are the nodes computed by processor P_3 locally. The number of computations, therefore, is $1 + 2 + 3 + \cdots + L/P = (L/P) + (L/P + 1)/2$.

Let's now consider P_2. The longest path from leaf nodes of this processor ends at level 4. We use the technique of triangles discussed above to calculate the total number of computation performed by processor P_2.

We superimpose all the local nodes of processor P_3 (represented as a triangle) to cover some of the processor P_2 nodes (that is, the first node (counting from the bottom) at the level 6 can be superimposed on to the third node at the same level, the third node at the same level). This way, we can cover most of the processor P_2 except for the square (actually a rhombus) at the bottom (consisting of first node at level 4, first and second nodes at level 5, and second node at level 6). The number of nodes on each side of this square is equal to L/P and hence, the total number of nodes in this square is $(L/P)^2$. Therefore, processor P_2 has $L/P + (L/P + 1)/2 + (L/P)^2$ total nodes for computations. Therefore, the number of total computations for P_i, $i = 0, \dots, P-1$ is $(L/P) \times (L/P + 1)/2 + (P - i - 1)(L/P)^2$.

Communication Recall that at each level, the higher numbered processor sends a data value to the neighboring lower numbered processor. Therefore, P_0 performs no remote communications. Processor P_3 sends $(L/P + 1)$ data values to processor P_2 and processor P_2 in turn sends $(L/P + 1) + L/P$ data values to processor P_1, etc. In general, processor P_i communicates $1 + ((L/P) \times (P - i))$ data values to P_{i-1}.

Therefore, total number of communication is

$$\sum_{i}^{P-1} \{(1 + L)/P \times (P - i)\} = (P - 1)(1 + L/P)$$

Nonrecombining algorithm In this algorithm, the master processor creates P leaves (for each asset in the case of the multi-asset derivative) and stores them in an array and distributes this array to distinct processors. The processors create their own subtree with the leaf nodes (of the master processor) as the root and perform the computations. At the end of the local computations, the processors send back the computed option values to the master processor in an array (for each of the each of asset tree in the case of the multi-asset derivative). Therefore, there are P sends and P receives by the master processor totaling $2P$ communications. The number of levels created by the master processor is $\log_2 P$. Therefore, to reach initially set number of levels L, each processor creates a subtree with $L - \log_2 P$ levels. The number of computation per processor is $2^{(L - \log_2 P + 1)} - 1$. Therefore, the total number of computation in the algorithm is $2^{(L+1)} - 1$.

4.4 Parallel multi-dimensional asset algorithm

The algorithms described above are for options with single underlying asset. As mentioned earlier, parallel algorithms are in great demand for financial problems, especially for long-dated options with many underlying assets. Financial derivatives with ten underlying assets are very common in the market place. We, therefore, extend the above algorithms for multi-dimensional derivatives, which have multiple assets as underlying components. This problem gives rise to immense effort in analyzing the effect of one of the underlying assets onto the other and the overall effect on the derivative itself. This leads to multi-dimensional analysis and in the simplest unoptimized form would require ten times more computing time.

As time periods and number of assets increase, the interactions among these assets would modify the computational pattern and this would further increase the computational complexity for such problems. The problem becomes computationally intractable due to many real-world constraints to be satisfied in pricing such derivatives. One of the other effects is on managing the portfolio [39], for which pricing of

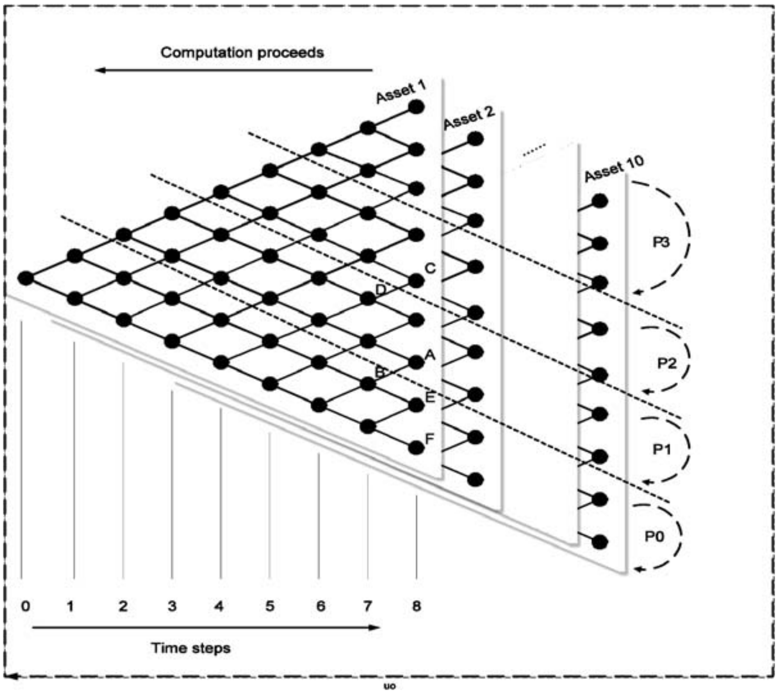


Fig. 5 Options with many underling assets

these derivatives form a fundamental problem that need to be solved accurately and expediently.

In our algorithm, the number of assets is fixed for a given experiment with an important assumption that they all have the same expiration time. This helps us to keep the computation time step constant for all the assets once we fix the number periods L into which the total period $[0, T]$ is divided into. This results in having a same tree structure for all the assets (Fig. 5). To handle the multi-assets, we introduced an array representing the option values of each of the assets. We modified the broadcasting function to enable array distribution among the processors. Each processor receives a portion of the tree.

Both algorithms have a number of variables. Type double is used for real values to ensure high accuracy of calculations. Since tree size may vary greatly, dynamic memory allocation is required. Using the input parameters read from a file, the other parameters are computed. A single processor does these readings and computations. Broadcasting is used to pass the read and calculated parameters to all processors. There are several different ways to broadcast data in MPI. One of them is the packing of all the data into a single variable and then broadcasting it to the processors. After receiving this variable, the processors unpack it into various parameters. We have found that this way is more efficient than the individual broadcasting of each parameter.

For example, the processor 0 works with the first n-nodes of the first asset tree, first n-nodes of the second asset tree, etc. The processors go simultaneously through the same portion of the each asset tree. In essence, we have kept the number of Send/Receive calls with multi-asset problem to be the same as the number of Send/Receive calls with single asset problem by folding the option values of the different assets into an array.

5 Developing mobile interface

There are five major platforms available for developing mobile applications such as BREW, Windows Mobile, Symbian, WAP (Wireless Application Protocol), and J2ME (Java 2 Micro Edition) [28]. For our research, we developed the architecture on a J2ME platform. J2ME is supported by major carriers (for example Nokia, Motorola) and it is relatively easy to deploy the application for a trader to download and install on mobile devices. We describe in this section J2ME and its implementation details of the current study. J2ME is a stripped down version of Java aimed at machines with limited hardware resources such as a PDA or a mobile phone [30]. The J2ME platform has two configurations: CDC (Connected Device Configuration) and CLDC (Connected Limited Device Configuration). The choice of configuration will depend on the memory constraints of the particular device. CLDC targets for devices with a constrained CPU and memory, which is generally a 32-bit CPU with 160 KB–512 KB memory. CLDC uses the Kilo Virtual Machine (KVM), a specialized virtual machine that supports only a limited memory. CDC is targeted for devices with more resources, usually a 32-bit CPU with more than 2 MB of memory. A CDC configuration layer runs on top of the C Virtual Machine (CVM). On top of these configurations, other profiles such as MIDP (Mobile Information Device Profile) and optional APIs can be layered in order to support user interface and other network functionalities.

In the following paragraphs, we provide details about three important design issues for J2ME that are essential for our study: (i) designing and building a MIDP User Interface (UI) (ii) communication of the MIDP and back end server and (iii) security issues of the network. MIDlet is a MIDP application [32]. Similar to applet, a MIDlet is a managed application. A web browser manages applets, whereas, the Application-Management System (AMS) manages MIDlet. Every MIDlet class handles its own logic and life cycle, which reflects the methods of the MIDlet class. There are three possible methods in a MIDlet's life-cycle such as startApp(), pauseApp() and destroyApp(). MIDlet enters the active state after the application manager calls startApp(); MIDlet remains in the active state until the application manager calls pauseApp() or destroyApp(). In the pause-App() method, MIDlet is temporarily suspended whereas in destroyApp(), the MIDlet completely terminates the application itself and awaits garbage 15 collection. In MIDP, UI classes are located in the javax.microedition.lcdui package of J2ME. In J2ME, commands are used to create UI objects that behave like buttons (action events in Java); commands such as OK, EXIT, and HELP are characterized by instances of command class.

Option pricing is computationally intensive. Since the required processing power and the memory are both in short supply on mobile devices, computation of option for particular asset is done on the server-end by utilizing the total functionality of

Java 2 Standard Edition (J2SE). Moreover, in order to optimize the consumption of resources on mobile devices, it is desirable to keep the communication to a minimum. Therefore, the connection between the server and the device is kept open just long enough to exchange user data (for example, name of the computational technique to be used to compute the option value together with the required accuracy).

The connection to dissimilar types of wireless devices will need different forms of connection interfaces. The Generic Connection Framework (GCF) [30] is available in J2ME/CLDC to reflect the need for small-footprint networking for a range of mobile devices. GCF is a hierarchy of interfaces defined in the "javax.microedition.io" package that allows mobile applications readily available to the trader on the network. The GCF interfaces reflect different capabilities and ensure the operations in a logical fashion. MIDP simplifies this GCF to a single connection type called HTTP (Hyper Text Transfer Protocol) and HTTPS (secure HTTP available in MIDP 2). HTTP is built around client requests and server responses, and it has two parts: header and content. The communication format (for example, XML, text, and binary) between MIDlets and the back end server in the body of HTTP depends on the design of the application. We tried with GET, HEAD, and POST methods, which are simple to implement and then with XML-RPC and KXML-RPC[2] over HTTP/HTTPS. In our random observations of speed and bandwidth tests, XML tends to have heavy bandwidth between the mobile and the server rather than byte arrays (either it is a string or data of any sort).

In order to provide enough security for data transmission, we will use secure HTTP (HTTPS) provided by MIDP 2.0 (If device support MIDP 2.0, it has default HTTPS; for example, Motorola E390 supports MIDP 2.0). On top of that, to provide additional security, we use open source lightweight API called the "Bouncy Castle" library that supports a large number of cryptography algorithms [32]. Therefore, the mobile component of our architecture will be secured. Finally, the task of deployment of the above application (MIDlet Suites) to a specific mobile device can be done using OTA (Over-The-Air installation of MIDlets) or Infrared (IR) or Bluetooth technology.

Before illustration of our server design, we will describe the design layout of our architecture with aid of diagrammatic representation.

6 Ubiquitous architecture for option pricing

In this architecture (Fig. 6), a broker remits a strike price and a contract period of a particular option (with a specific asset) to a subscribed Mobile/PDA/wireless trader. The information provided by the broker is incomplete and can only partially aid in computing the fair value of the option. If the trader (client) needs to decide if entering the option contract is beneficial, the trader has to compute the option value by selecting the *underlying assets* and *computational technique* to be used for computation at the back end server. Moreover, the trader enters the *number of time steps* to be processed for computing the option price. All the above values entered by the trader are sent to the web/compute server for computation.

[2]XML-RPC is a standard way of implementing remote procedure calls (RPC) using XML and HTTP. To accomplish this, it uses XML to mark up all of the method and uses HTTP to transfer the methods.

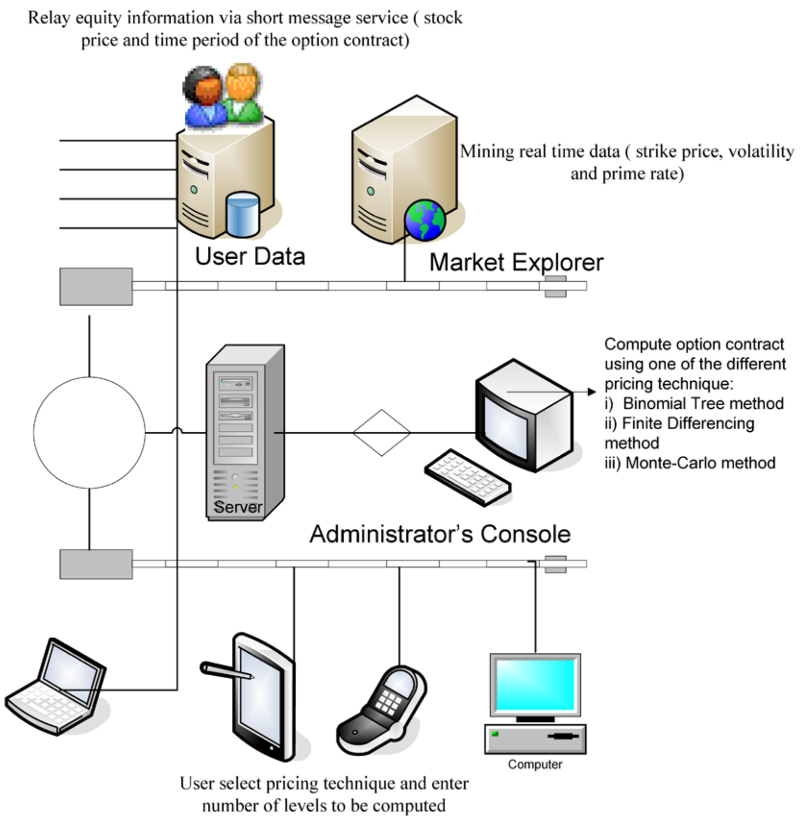

Relay equity information via short message service (stock price and time period of the option contract)
User Data
Mining real time data (strike price, volatility and prime rate)
Market Explorer
Compute option contract using one of the different pricing technique:
i) Binomial Tree method
ii) Finite Differencing method
iii) Monte-Carlo method
Server
Administrator's Console
Computer
User select pricing technique and enter number of levels to be computed

Fig. 6 Mobile infrastructure

Server and its Functional Operations: Once the client submits time steps, underlying asset and the computational technique, back end servlet mines (using table mining technique) real-time values such as spot price, volatility, and prime interest rate from the web sources. The server then computes option price with the above real-time values, using finite-differencing technique described in Sect. 4, or other computational techniques such as binomial tree or Monte Carlo method.

Theft and misappropriation are greatest vulnerable factors in wireless trading. Mobile devices can be easily stolen and misused which may result a financial debacle. Consequently, access control and identification of authentic trader are undoubtedly vital in wireless trading. In our framework, transmission mode between client and server are secured in every aspect (as mentioned in Sect. 5). In addition, in order to access his/her portfolio, a trader has to setup and will able to access their accounts that will facilitate customized tables and data analysis based on the underlying computation (for example, to access tables such as risk-free zone, healthy bids, and favored stocks which are discussed in results section). Trader's devices are enabled by security token and each token generates a 5-digit security code that is periodically changed, updated, and acknowledged by the trader to the server or vice versa.

7 Architecture results

In this section, we report our experimental results of the architecture. We have divided our results into two parts: (A) computational results (B) mobile-enabled option trading scenario.

Computational results In Fig. 7, we present one set of results on the computed call option values. We notice that as the strike price increases the call option value decreases, as expected. Validations of these results are done by simple manual calculation of a smaller binomial tree with identical parametric conditions. Moreover, for the current study, accuracy on option value computed is to the fourth decimal place. This accuracy is sufficient for an academic exercise. However, we can obtain more accurate results, which will incur higher computational cost. Implementing the computational techniques in a parallel environment [24] will circumvent the computational cost. This is not the objective of the current study, however.

Intel corp (*INTC*) *CALL option* We calculated option values for INTC CALL online with varying strike prices (for both actual and speculated values) and stock prices. Table 1 presents the option values computed at various stock and strike prices. The real time data for S is 25.52. To make some speculation, we have computed the option values for stock, strike prices around the currently available stock, and strike prices.

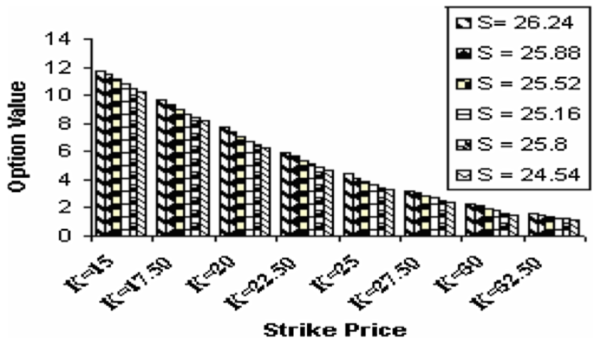


Fig. 7 Option values at various stock and strike prices

Table 1 INTC (CALL) option values for varying stock and strike prices

	$S = 26.24$	$S = 25.88$	$S = 25.52$	$S = 25.16$	$S = 25.8$	$S = 24.54$
$K = 15$	11.77553	11.44067	11.1058	10.77094	10.43608	10.19423
$K = 17.50$	9.670142	9.345136	9.0259	8.706682	8.387466	8.156904
$K = 20$	7.720975	7.401749	7.0826	6.763296	6.44406	6.213517
$K = 22.50$	5.945113	5.67445	5.40381	5.133154	4.362501	4.667031
$K = 25$	4.413729	4.143077	3.87243	3.601772	3.383281	3.255619
$K = 27.50$	3.21096	3.034197	2.85743	2.680671	2.503908	2.376246
$K = 30$	2.33158	2.154825	1.979806	1.801299	1.624536	1.502361
$K = 32.50$	1.544512	1.457355	1.371199	1.284542	1.197885	1.134185

Table 2 Healthy bids for option contract (INTC) when stock is 25.52

Strike price	ASK-price	Option price	Error-R%	Healthy BID
15	11.4	11.1058	2.50%	11.25.29
17.5	9.7	9.0259	6.94%	9.3559
20	7.9	7.0826	10.34%	7.4913
22.5	6.2	5.40381	12.84%	5.8125
25	4.8	3.87243	19.00%	4.33613
27.5	3.6	2.85743	20%	3.05099
30	2.6	1.979806	NA	NA
32.5	1.85	1.3711989	NA	NA

This is done to come up with a healthy bid to enter the option contract as presented in Table 2.

Risk-free zone To be in the risk-free zone, we have set up a healthy bids based on the ASK price and error rate of on-line contracts with stock price 25.52. *Ask price* can be defined as the price at which a writer is willing to sell (buy) an asset; also called the offer price. We propose a four-step procedure for calculating "healthy bids" of the option contract. (1) Ask price is mined from on-line web sources (for example Yahoo!) and option price is calculated (as mentioned in Sect. 3) based on real time values. Once we have option price of the contract and on-line ASK value, we can calculate percentage error, mean error, and finally healthy bid. (2) Calculating percentage error*: Percentage Error rate* can be calculated as: (ASK price − Option price)/ASK price × 100.

For our research, we consider errors within the range of (0–10%). If the error rate is more than the specified range, it is discarded. For practical purposes, the real-time error could be improved with advanced computational techniques mentioned earlier. If error is within the specified range, we can continue with the calculation of the *healthy bid*. (3) Mean error: Healthy bid can be calculated based on the mean error rate and is calculated by the formula: abs((ASK price + Option price)/2 − ASK price).

(4) If (ASK price − Option price) < 0: Healthy bid = Ask − mean error and if (ASK price − Option price) > 0: Healthy bid = Option price + mean error. As seen in Table 1, the trader will be provided with various tables with speculated stock values to provide various scenarios of option values and a healthy bid. These tables are made available to the trader (client) from the web/compute server. Depending on the current knowledge of the trader on the behavior of the underlying asset, the trader will be able to select one of the healthy bids (please see Table 2—an example for Intel call option) and instruct the broker to enter the option contract at that bid. If the writer finds this bid comfortable, he/she will agree to this price and will agree to sell the underlying asset at the agreed upon strike price at the maturity date. In essence, the investor (client), therefore, has used his/her mobile device ubiquitous to value an option and enter the contract with a level of comfort that the investor can expect a profit from the option contract.

Table 3 Yield point of various companies' CALL contracts

Symbol	Strike price	Stock price	Historic volatility	Option value	G_i
MSFT	25.5	25.48	15.56%	3.137521933	4.50%
YHOO	30	37.41	24.19%	10.7066896	16.44%
MNT	45	41.53	27.78%	5.786570944	11.54%
SUN	105	108.68	25.94%	20.24165532	NA
GOOG	230	284.84	31.95%	1.121790707	0.60%

Table 4 Yield point of various companies' PUT contracts

Symbol	Strike price	Stock price	Historic volatility	Option value	G_i
MSFT	30	25.48	15.56%	4.52	6.55%
YHOO	35	37.41	24.19%	2.154014463	3.36%
MNT	45	41.53	27.78%	6.074323101	1.12%
SUN	105	108.68	25.94%	8.398909621	NA
GOOG	280	284.84	31.95%	31.6684338	17.49%

Favored stock To find out the preferred stock among various sectors, we have introduced G_i factor, which is based on yield point. G_i is calculated using a formula (Tables 3 and 4 for calls and puts on various stocks): $G_i = (c_i \times 100)/(S_i - \text{Mean } S)$ where c_i is the option price; S_i strike price; $\text{Max}(G_i)$ is the favored stock.

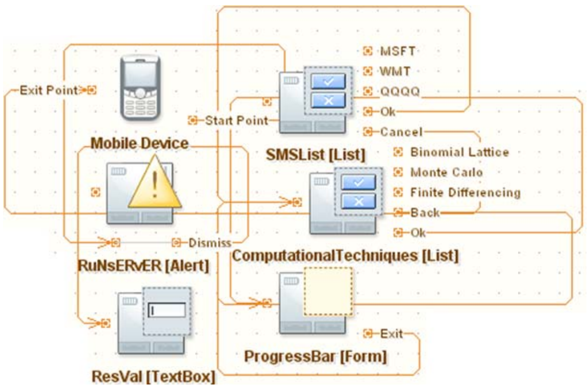
7.1 Mobile-enabled option trading scenario

Mobile emulation is done on Net Beans, which is open source software with integration of J2ME Wireless Toolkit platform. Moreover, UEI (Unified Emulator Interface) compatible emulators allow us choosing different devices (for example, NOKIA and MOTOROLA) from various companies [31]. The following Mobile screens (1–5) describe the flow design and the trading scenario. This architecture enables pricing multiple stock options with various strike prices in ubiquitous fashion. The contract/pricing information of particular stock is updated continuously to the trader on the move by aid of floor-services (exchanges) or brokers in time-to-time fashion. Once the clients get the option price from the web server, he/she will utilize built-in wireless device's small computing power to calculate healthy bids for various stocks in different sectors and simultaneously choose favored stock among them.

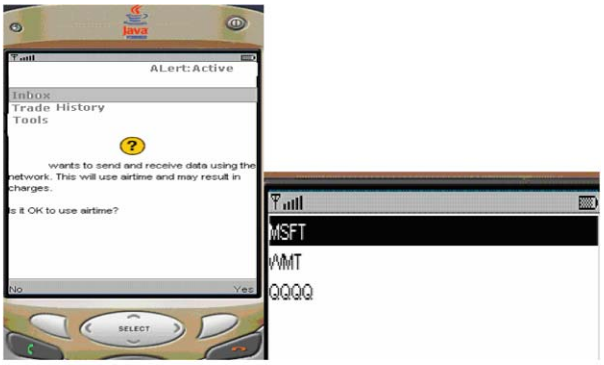
8 Conclusions

Fundamental challenges to design an ubiquitous software architecture for an option trader that is addressed in this paper emanate from three different domains: option trading, web mining, and mobile computing. Every domain has its own challenges for its functionality. We list the challenges that we faced to build the architecture and

Screen 1 Flow design of mobile terminal and server response



Screens 2 and 3 Alerts from the trading floor-services or brokers; and active stocks on the trading floor


ALert:Active
Inbox
Trade History
Tools

wants to send and receive data using the network. This will use airtime and may result in charges.

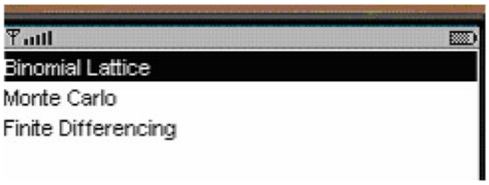
Is it OK to use airtime?

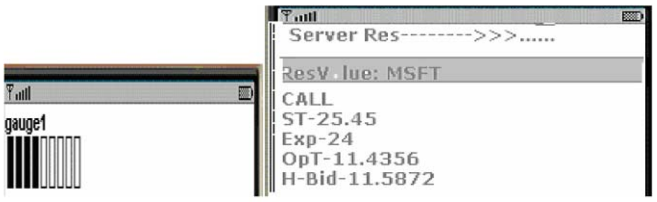
No Yes

MSFT
WMT
QQQQ

(2) (3)

Screen 4 Details of the contract and computational techniques available for pricing


Binomial Lattice
Monte Carlo
Finite Differencing


gauge1

```
Server Res-------->>>......
ResV lue: MSFT
CALL
ST-25.45
Exp-24
OpT-11.4356
H-Bid-11.5872
```

(5) (6)

Screens 5 and 6 Response of the computed results from the web server (back end servlet) our solution to each of the respective challenges, and thereby our contribution to the emerging mobiquitous business scenario:

Computation issues (i) option pricing by itself is a computationally intensive problem; (ii) parameters required for computing option price are continuously changing due to market fluctuations. In such a situation, accurate results will depend heavily on appropriate use of current market conditions; (iii) configuring and implementing the existing/new computational algorithms and handling multiple traders simultaneously and remotely is a challenging task.

Contribution We have parallelized and implemented couple of computational techniques (binomial tree method and finite-differencing technique) maintaining accuracy to a large extent. Higher accuracy can be obtained by introducing appropriate new technique (when and if available or developed) in this module of our architecture.

Challenges in web mining Mining real-time finance data from reliable web sources and forwarding the observed results.

Contribution Heuristics for single dimensional and two-dimensional tables are employed that are simple yet significant, however, not described in the current version due to lack of space.

Challenges in mobile computing (i) designing and building mobile trading terminal for computing option price any *time* and *any where* is one of the main challenging components of the current research; (ii) as the architecture depends heavily on real time data access, network connectivity, and security between mobile client and back end server remain an important issue.

Contribution To handle these issues, we have employed iterative secured-flow-design-approach for building screen logic and layout design of the Mobile interfaces. In addition, to provide additional security to wireless devices, we employed lightweight API called “Bouncy Castle.”

The essence of this framework is in its novelty, which stems from three different domains to enable mobile trader to compute the price of an option in pervasive fashion, which is an important application in option trading. Hence, this is one of the first studies that facilitate the service of option pricing on-the-go. For future work, we will introduce option sensitivities such as delta, gamma, vega, and others that are widely used by active traders to compute the exposure of portfolios that contain options. Each of these Greek measures show how the portfolios respond to a change in some variables (maturity period, volatility, strike price, etc.). Conclusively, our architecture enables an active investor to price options in real-time using various computational techniques in mobiquitous fashion that would pave way for effective investment trading toward higher profitability.

Acknowledgements The last two authors acknowledge partial financial support from the Natural Sciences and Engineering Research Council (NSERC) Canada through *Discovery Grants* and to the University Research Grants Program (URGP) of the University of Manitoba.

References

1. Hull JC (2002) Options, futures and other derivatives, 5th edn. Prentice Hall, Englewood Cliffs
2. Varshney U, Vetter RJ (2002) Mobile commerce: framework, applications and networking support. MONET 7(3):185–198
3. Varshney U (2003) Location management for mobile commerce applications in wireless internet environment. ACM Trans Internet Technol 3(3):236–255
4. Lyytinen K (2001) Mobile commerce: a new frontier for E-business. In: Proc 34th annual Hawaii international conference on system sciences, vol 9, January 2001, p 3509
5. Sadeh N (2002) M-commerce: technologies, services, and business models. Wiley, New York
6. Sheshagiri M, Sadeh N, Gandon F (2004) Using semantic web services for context-aware mobile applications. MobiSys 2004 workshop on context awareness, Boston, June 2004
7. Sun S, Su C, Ju T (2005) A study of consumer value-added services in mobile commerce: focusing on domestic cellular phone companies in Taiwan, China. In: Proceedings of the 7th international conference on electronic commerce, Xi'an, China, vol 113. Assoc Comput Mach, New York, pp 597–606
8. Geoffrey E, Phillips N (2004) Mobile commerce and wireless computing systems, 1st edn. Addison-Wesley, Reading
9. Kargupta H, Park B, Pittie S, Liu L, Kushraj D, Sarkar K (2002) Mobimine: monitoring the stock market from a PDA. ACM SIGKDD Explor 3(2):37–46
10. Ajenstat J, Jones P (2004) Virtual decision making for stock market trading as a network of cooperating autonomous agents. In: Proceedings (CD-ROM) 37th annual Hawaii international conference on system sciences, Big Island, Hawaii, January 2004
11. Roche J (2002) Mobile trading comes of age. In: United Nations conference on trade and development, September 2002
12. Roche J (1999) The global trading village, mutual benefits for the international online trading community and developing countries. In: United Nations conference on trade and development, April 1999
13. Roche J (2000) WEB to WAP: for a real start of the new millennium. United Nations conference on trade and development, October 2000
14. FinTools (2000) Options calculator. http://www.fintools.com, Montgomery Investment Technology, March 2000
15. Windale Technologies (2005) Option pricing analysis X software. http://windale.com/optionsx.php3, June 2005
16. FIS Group (2003) Option calculator. www.fis-group.com
17. CommSec (2003) Keep up the market, wherever you are. CommSec. http://www.satisfac.com.au/convCommSecsharetrading.htm, September 2003
18. QuoTrek (2003) Real time on-line data. http://www.quotrek.com/default.asp, February 2003
19. Microsoft mobile solution partner, pit trader work wirelessly with pockets PCs and solution from PhatWare. Leapfrog Technologies, Mobility Partner Council, San Diego, CA, 2003
20. Web A (2004) The tools of the trade. Incisive Media Investments Limited, Hedge Fund Review, London
21. Chang E, Cheng K (2003) Stock options to complement stock trading. Technical report, Hong Kong Exchanges Clearing Limited—Educational Article, No 18, October 2003
22. Thinkorswim Inc (2006) Stock option trading: online brokerage services. http://www.thinkorswim.com/tos/displayPage.tos?webpage=softwareLobby, March 2006
23. Kola K, Thulasiram RK (2005) UNNATI: ubiquitous option pricing—a finance application framework. In: Proceedings of the 13th international conference on advanced computing and communications, Coimbatore, India, December 2005, pp 191–198
24. Chhabra A, Thulasiraman P, Thulasiram RK (2004) Distributed computing with CORBA and multithreading with Java OpenMP on NoCs: a framework for CFD. In: Proceedings (CD-ROM) 12th international conference on advanced computing and communications, Ahmedabad, India, December 2004
25. Barua S, Thulasiram RK, Thulasiraman P (2005) High performance computing for a financial application using fast Fourier transform. In: Proceedings of European parallel computing conference (EuroPar 2005), Lisbon, Portugal. Lecture notes in computer science, vol 3648. Springer, Berlin, pp 1246–1253

26. Rahmail S, Shiller I, Thulasiram RK (2004) Different estimators of the underlying asset's volatility and option pricing errors: parallel Monte Carlo simulation. In: Proceedings international conference on computational finance and its applications, Bologna, Italy, April 2004, pp 121–131
27. Thulasiram R, Litov L, Nojumi H, Downing C, Gao G (2001) Multithreaded algorithms for pricing a class of complex options. In: Proceedings (CD-ROM) of the IEEE/ACM international parallel and distributed processing symposium, April 2001
28. Eckstein R, Topley K (2002) J2ME in a nutshell. O'Reilly, Paris
29. Mallick M (2003) Mobile and wireless design essentials. Wiley, New York
30. White J (2001) An introduction to Java 2 Micro edition (J2ME); Java in small things. In: Proceedings of the 23rd international conference on software engineering, Toronto, Ontario, Canada, pp 724–725
31. Kaisa N, Mattila V, Ruuska S (1999) Design: designing mobile phones and communicators for consumer needs at Nokia. Interactions 6(5):23–26
32. Itani W, Kayssi A (2004) J2ME application-layer end-to-end security form-commerce. J Netw Comput Appl 27(1):13–32
33. Tannehill JC, Anderson DA, Pletcher RH (1997) Computational fluid mechanics and heat transfer, 2nd edn. Routledge, London
34. Thulasiram RK, Zhen C, Chhabra A, Thulasiraman P, Gumel A (2006) A second order L_0 stable algorithm for evaluating European options. Int J High Perform Comput Networking 4(5/6):311–320
35. Wilmott P, Howison S, Dewyenne J (1995) The mathematics of financial derivatives. Cambridge University Press, Cambridge
36. Tavalla D, Randall C (2000) Pricing financial instruments: the finite difference method. Wiley, New York
37. Thulasiram RK, Downing CT, Gao GR (2000) A multithreaded parallel algorithm for pricing American securities. In: Proceedings (CD-ROM) of the international conference on computational finance, London, England, June 2000
38. Duffie D (1996) Dynamic asset pricing theory. Princeton University Press, Princeton
39. Haugh MB, Lo AW (2001) Computational challenges in portfolio management-tomorrow's hardest problem. Comput Sci Eng 3(3):54–59
40. Black F, Scholes M (1973) The pricing of options and corporate liabilities. J Polit Econ 81:637–654
41. Cox JC, Ross SA, Rubinstein M (1979) Options pricing: a simplified approach. J Financ Econ 7:229–263

Kiran Kola graduated in 2002 with a Bachelors in Engineering (Computer Science) from the University of Madras, Chennai, India. He successfully completed his M.Sc. program in Computer Science at University of Manitoba, Canada in 2007. For his M.Sc. thesis he designed and developed a three tier architecture that includes client (mobile, pc, and laptop), server (compute server) and database (market explorer and customer related data) for the financial trading applications especially for options trading. He received the best paper award for part of his thesis work at the Intl. Symp. on Parallel and Distributed Processing and Applications held in December 2006, Sorrento, Italy. His current research interests are in Autonomic Computing, Mobile trading platforms and advancements in Supercomputing.

Ruppa K. Thulasiram (Tulsi) is an Associate Professor with the Department of Computer Science, University of Manitoba, Winnipeg, Manitoba. He received his Ph.D. from Indian Institute of Science, Bangalore, India and spent years at Concordia University, Montreal, Canada; Georgia Institute of Technology, Atlanta; and University of Delaware as Post-doc, Research Staff and Research Faculty before moving to University of Manitoba as an Assistant Professor. Tulsi has undergone training in Mathematics, Applied Science, Aerospace Engineering, Computer Science and Finance during various stages of his education and post doctoral training. Tulsi's current primary research interests is in the emerging area of Computational Finance. Tulsi has developed a curriculum for cross-disciplinary computational finance course at University of Manitoba and currently teaching this at both graduate and undergraduate level. His other research interests include Scientific Computing, Ad-hoc Networking for M-Commerce applications, and Mathematical Finance.

He has published number of papers in the areas of High Temperature Physics, Gas Dynamics, Scientific Computing and Computational Finance in leading journals and conferences. Tulsi has been serving in many conference technical committees related to parallel and distributed computing, Neural Networks, Computational Finance and has been a reviewer for many conferences and journals. He has started a workshop "Intl. Workshop on Parallel and Distributed Computing in Finance" in conjunction with the IPDPS. He is a member of the ACM, IEEE and SIAM societies.

Parimala Thulasiraman received B.Eng. (Honors) and M.A.Sc. degrees in Computer Engineering from Concordia University, Montreal, Canada and obtained her Ph.D. from University of Delaware, Newark, DE, USA after finishing most of her formalities in McGill University, Montreal, Canada. She is now an Associate Professor with the Department of Computer Science, University of Manitoba, Winnipeg, MB, Canada. The focus of her research is on the design, development, implementation and performance evaluation of scalable parallel algorithms for multicore heterogeneous processors in computational science applications such as computational biology, computational finance, medical imaging or computational medicine on advanced architectures. Over the past few years, she has moved towards developing distributed algorithms for wireless networks using nature inspired algorithms such as Ant Colony Optimization techniques. She has published several papers in the above areas in leading journals and conferences and has graduated many students. Parimala has organized conferences as local chair, program chair and tutorial chair. She has has been serving as a reviewer and program committee member for many conferences. She has also been a reviewer for many leading journals. She is a member of the ACM and IEEE societies.