Financial Application as a Software Service on Cloud

Saurabh Kumar Garg, Bhanu Sharma, Rodrigos N. Calheiros, Ruppa K. Thulasiram*, Parimala Thulasiraman**, and Rajkumar Buyya

Department of Computing and Information Systems
University of Melbourne
Melbourne, Australia
{sgarg,raj}@csse.unimelb.edu.au,
{bsharma}@cs.umanitoba.ca,
rodrigo.calheiros@gmail.com, {tulsi,
thulasir}@cs.umanitoba.ca

Abstract. In this work, we propose a SaaS model that provides service to ordinary investors, unfamiliar with finance models, to evaluate the price of an option that is currently being traded before taking a decision to enter into a contract. In this model, investors may approach a financial Cloud Service Provider (CSP) to compute the option price with time and/or accuracy constraints. The option pricing algorithms are not only computationally intensive but also communication intensive. Therefore, one of the key components of the methodology presented in this paper is the topology-aware communication between tasks and scheduling of tasks in virtual machines with the goal of reducing the latency of communication between tasks. We perform various experiments to evaluate how our model can map the tasks efficiently to reduce communication latency, hide network latency ensuring that all virtual machines are busy increasing response time of users.

1 Introduction

Cloud computing offers on-demand software service to customers without having the customer know the inner details of the software or IT infrastructure used for their service. Businesses outsource their computing needs to the Cloud to avoid investing in infrastructure and maintenance. The charges they pay to Cloud providers are generally seen as economical while considering the cost that they would incur having the infrastructure in-house. One sector that can significantly benefit from Cloud is the financial sector.

* The author was visiting from Department of Computer Science, University of Manitoba, Canada; Also author for correspondence: tulsi@cs.umanitoba.ca

** The author was visiting from Department of Computer Science, University of Manitoba, Canada

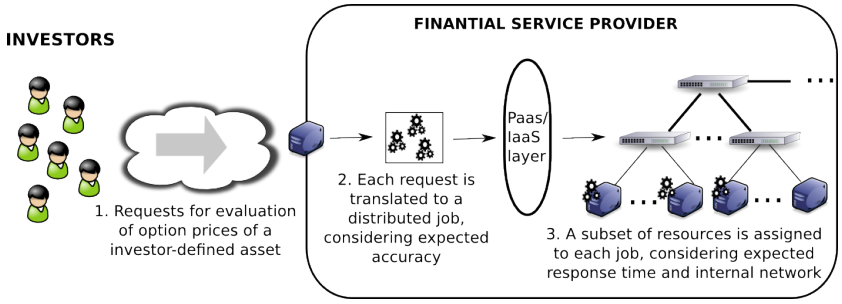

Fig. 1. Financial Service Provisioning in the Cloud.

For example, an investor would be interested in obtaining information that would help in making decision whether to buy a stock at a future date, at a particular price based on some basic information available in public such as Yahoo!Finance. This kind of future investment is referred to as a *option*. This requires knowledge of the current stock price, stock volatility and the proper time to exercise the stock for profit. There are many algorithms [5, 7, 3] to *price an option*. In finance, this problem is called an *option pricing problem*.

Investors interested in pricing an option would need to have working knowledge of these algorithms to help them take an informative decision on computing the option prices. The algorithms used in the option pricing problem are computationally intensive and require parallel processing to obtain results in real time. These computations are very complicated for an investor who is not familiar with the algorithms. The option pricing problem, falls under the category high performance computing applications and several works have already been done in this area [12].

In this study, we propose a SaaS model that provides service to ordinary investors, unfamiliar with finance models, to evaluate an option that is currently being traded before taking a decision to enter into a contract. This provider owns the data center infrastructure and uses it to host the Financial SaaS.

In our SaaS Model (Figure 1), decision on the kernel (option pricing algorithm) to be deployed for a user request, number of tasks to be generated for such request, number of resources to be assigned to the request, and which resources to serve the request is performed at the PaaS level: SaaS requests received from users are forwarded to the PaaS component that make such decisions, which in turn deploys the tasks to the available infrastructure (IaaS). Note that if the SaaS service were offered by a provider that does not own the infrastructure, a fine control over the platform that enables better QoS for investors would be hard to achieve.

To our knowledge no work exists that addresses the need for decision making service involving financial instruments on Cloud. Contributions of the paper are: (a) Making decision on a particular financial model to use that would best fit the user's requirements and constraints. (b) Satisfying investors constraints on time deadline and accuracy. (c) Mapping the tasks to appropriate VMs considering various latencies. (d)

Optimizing the number of request processed per second considering HPC nature of the particular algorithm being considered for option pricing.

We organize the rest of the paper as follows. In section 2 we discuss financial option and task scheduling on Cloud. In Section 3, we describe our system model and in Section 4 we present our algorithm for evaluation of the algorithms on Cloud. In Section 5 we describe the results and conclude in Section 6.

2 Background and Related Work

In this section, we discuss financial options and a literature survey of task scheduling on Clouds.

Financial Options: Formally, an *option* is a contract in which the buyer (generally known as the option *holder*) of an option has the right but without any obligation to buy (with *call* option) or sell (with *put* option) an underlying asset (for example, a stock) at a predetermined price (*strike price, K*) on or before a specified date (*expiration date, T*). The seller (known as *writer*) has the obligation to honor the terms specified in the option contract. The holder pays a premium to the writer (see for example [8]). An European option can be exercised only at the expiration date whereas an American option may be exercised on any date before the expiration date. We have considered four different algorithms in this study for implementing on the CSP side to render option pricing services: Binomial Lattice [7, 13], Monte-Carlo simulation [3], Fast Fourier Transform [5] and Finite-Difference technique [14].

Task Scheduling on Cloud: A key component of the methodology presented in this paper is the topology-aware communication between tasks and scheduling of tasks in VMs with the goal of reducing the latency of communication between tasks. Kandalla *et al.* [9] proposed topology-aware algorithms for communication between MPI tasks. However, this approach does not consider virtualized data centers as the underlying hardware infrastructure supporting the application and therefore, cannot be directly applied in our proposed solution. Volckaert *et al.* [15], proposed a network-aware task scheduling algorithm on grids following an embarrassingly parallel (bag of tasks) approach, which is not applicable to financial applications that introduce communication overhead between tasks. Lee *et al.* [10] proposed a topology-aware resource allocation mechanism for IaaS Clouds using genetic algorithm to find the optimal placement of tasks to machines which again follows an embarassignly parallel approach. Coti *et al.* [6] proposed topology-aware scheduling of MPI applications on Grids which cannot be directly applicable to Cloud.

3 System Model

Customers of SaaS provider (CSP) are investors who need to evaluate (the price of) an option that is currently being traded before taking a decision to enter into a contract or not. Our proposed system model will be able to provide information as whether the entering the option contract could be profitable. For this purpose, the CSP in our model

uses one of the four algorithms for option pricing: fast Fourier transform (FFT), finite-difference (FD), binomial lattice, and Monte-Carlo simulation. Each of these algorithms is representative of typical HPC applications with their different computation and communication needs. The customers' request for resource contains: service description, required accuracy, and deadline for service response. Each of the option pricing algorithms has different processing times which affects service deadline. Also, each algorithm provides an accuracy different from another. Therefore, customers are charged for the service depending on the required accuracy and service deadline. Each customer may have their own parameters or input for an option. For example, the current stock price (S), strike price in the contract (K), volatility of the asset (σ), expiration time in years T, and current interest rate (r). These parameters are provided as service description to the CSP. Another important parameter is the number of time steps N. For example, in the binomial lattice algorithm, increasing the number of time steps, makes the problem more fine-grained providing more accurate results. However, this increases processing time. The CSP executes customers' requests at its own data center. With this

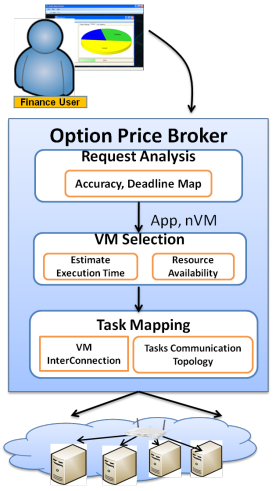

Fig. 2. System Model.

model, the service provider can ensure the secrecy of the models they use to provide service to their customers, which can be more accurate than models available with other CSP.

The system architecture that can support the described scenario is depicted in Figure 2, and contains two main components, namely *option price broker* and *data center*.

The option price broker receives user requests, analyzes them, and submits them to the data center. It keeps information about resource availability and number of requests being processed. It processes each request in three distinct phases: (i) defines the preferred order of execution of algorithms based on request's accuracy and deadline; (ii) searches for a mapping of tasks to VMs considering the preferred order of

algorithms: that is, if it is unable to accommodate a set of VMs for its first choice algorithm, it evaluates the possibility of the second algorithm; (iii) maps the tasks based on their communication requirement, on the VM's network topology, and on memory requirements.

The data center handles the actual execution of tasks that represent user requests. The data center network topology considered in this paper assumes servers placed in racks are connected using edge and aggregation switches and core routers. In this paper, we assume static routing inside the data center. The fat tree topology on which the servers are organized can be over-subscribed depending on the number of servers connected to the edge switch. Moreover, each server hosts 2^k VMs where k can be predetermined by the provider.

4 Algorithm and Evaluation

In this section, we discuss the mechanism for processing user requests. Each request has the form $(OptVariables_i, A_i, d_i)$, where $OptVariables_i$ are the variables for calculating option pricing, A_i is the required accuracy and d_i is the service deadline by which user is expecting the results. The option price broker performs three phases of computations:

4.1 Phase 1: Algorithm Decision

In this step, the broker decides which algorithm to use to service the customers request. This depends on the desired accuracy and service deadline.

Table 1. Rank of techniques for accuracy requirements.

	Low (0-50%)	Medium (50-70%)	High (70-100%)
Binomial	2	1	3
Monte-Carlo	1	2	4
Finite Diff.	3	3	1
FFT	4	4	2

To determine which algorithm provides better accuracy, the CSP considers a precomputed table, derived from benchmark results of the algorithms. In Table 1, each of the algorithm used in this study is ranked (1 is highest and 4 is lowest) according to three accuracy levels: low, medium and high. This is derived from our previous experience [11] of running these algorithms on different platforms. Similarly, Table 2 depicts the ranks of these algorithms in terms of the execution time. For example, if a customer requires high accuracy but has a relaxed (large) deadline, then finite-difference technique is chosen as the preferred algorithm to compute the option prices. However, if a user needs high accuracy and has a tight deadline, the request is rejected. Note, that the four algorithms are ranked based on the request characteristics and this is the order that will be considered.

Table 2. Rank of techniques for timing requirements.

	Low (quicker)	Medium (moderate)	High (large)
Binomial	1	2	4
Monte-Carlo	2	1	3
Finite Diff.	4	4	1
FFT	3	3	2

Since these algorithms are also communication intensive, some thought needs to be given to the communication structure of the algorithms to better schedule them on the VMs. For FFT, we follow the Cooley-Tukey butterfly algorithm [2]. The parallel FFT algorithm requires $logP$ communications and $logN - logP$ computations. Therefore, the execution time of each task is given by $t_{computation} * (logN - logP) + t_{communication} * logP$, where N is number of elements (or time steps) and P is the number of VMs.

The option price broker also makes a decision on the number of VMs required to execute tasks for ensuring its completion within the deadline.

4.2 Phase 2: Virtual Machines Selection

In this phase, the option price broker decides on the set of VMs needed to execute the algorithm selected in phase 1. As mentioned earlier, each server in the data center hosts 2^k VMs. Moreover, each algorithm has communication and computation needs that have to be met by the internal network topology and server resources, respectively. Since the data center runs several requests/algorithms simultaneously, the broker selects a set of VMs for an incoming request in such a way that the communication overhead is minimized. For example, if FFT is being considered to be executed, the broker tries to select VMs that belong to the same server or whose servers are positioned in the same rack. This ensures data locality. Therefore, the broker adopts the following strategy for VM selection. Let the number of VMs required by the request i for given accuracy level A_i and deadline d_i, be $numVmReq_i$. This is assumed to be a power of two. Let $EstExTime_i$ be the estimated execution time of each task. The following mapping strategies are used to schedule different algorithms.

For FFT, tasks have to be confined to the same server or rack. The selection of VMs for this algorithm is described in Algorithm 1. For Finite-Difference technique, since communication is only between neighbouring taks, there is no restriction about VM placement, and the mapping is described in Algorithm 2. Finally, in the case of Monte-Carlo and Binomial Lattice, that are loosely-coupled algorithms, tasks are distributed in a round-robin strategy.

4.3 Phase 3: Mapping tasks to selected VMs

In this phase, the actual scheduling of each task from various applications to VMs takes place, following the decisions made in the previous phases. The mapping considers communication between tasks.

Algorithm 1: Mapping of tasks to VMs in the FFT algorithm.

```
numHostReq_i = numVmReq_i/2^k;
foreach edge switch do
    Search numHostReq_i available servers that can process the job within deadline;
    if servers are found then
        map tasks to VMs based on network topology required for communication;
        exit;
    end
    else
        search for numVmReq_i available VMs in a same rack;
        if VMs are found then
            map tasks to VMs based on network topology required for communication;
            exit;
        end
        else
            try the next algorithm in the priority list;
        end
    end
end
```

Algorithm 2: Mapping of tasks to VMs in the FD algorithm.

```
numHostReq_i = numVmReq_i/2^k;
foreach edge switch do
    search for numVmReq_i available VMs in a same rack;
    if VMs are found then
        map tasks to VMs based on network topology required for communication;
    end
    else
        search for numHostReq_i servers that can process the job within deadline;
        if servers are found then
            map tasks to VMs based on network topology required for communication;
        end
        else
            try the next algorithm in the priority list;
        end
    end
end
```

5 Results and Discussions

This section presents the performance results of our proposed algorithm to enable Option Pricing as Cloud Software service. We simulated a SaaS Cloud scenario using CloudSim [4]. We compare our mechanism to a general approach of deploying the VMs across a data center without any knowledge of network and application requirements. We call this approach as $BlindMethod$. In phase 2 using this mehtod, tasks are randomly assigned to VMs. The simulation design reflects configurations that are similar to actual data centers and can determine the performance of our proposed mechanism (computation strategy).

5.1 Option Pricing SaaS provider's configuration

Current Cloud data centers generally contain commodity hardware with hierarchical tree network topology [1]. Therefore, in our experiments each simulated server is equivalent to a Intel Core i7-920 with 4 cores and 8 threads (virtual cores), 2.66 Ghz with 8 GB RAM. We have deployed two VMs per server each with 4 cores and 4 GB RAM. Each server is connected to the edge network with 1 Gigabit Ethernet link. Servers are placed in racks, and each rack has one edge switch. We have one 4-port root switch, four 2-port aggregate switch with one up link and eight edge switches. The edge and aggregate switches are connected by 10 Gigabit Ethernet links. The network bandwidth between aggregate switches is 20 Gbps. The default number of servers in each rack is 10, therefore 160 VMs are simulated. The switching delays for different number of hops are based on the work by Kandalla et al. [9], i.e. intra rack communication delay is $1.57\mu sec$, delay due to communication through aggregate switch is $2.45\mu sec$ and delay due to communication through root switch is $2.85\mu sec$. To estimate the execution time of different algorithms we run different set of experiments using all four option pricing algorithms on a Intel Core i7-920 4cores/8thread (virtual cores), 2.66 Ghz with 8GB RAM machine.

5.2 Option Pricing Request Generation

Since there is no trace available from data centers running financial applications, we generated different types of requests using uniform distribution for varying option pricing inputs, accuracy (low, medium, high) and deadline (low, medium, high) requirements. The arrival rate of customer requests is 10000 requests per second.

5.3 Performance Metrics and Experimental Scenarios

We use two metrics to evaluate our computation strategy: average processing time and network overhead. The average processing time indicates how fast our mechanism can process user requests, which is an important quality of service metric for any SaaS provider. The network overhead indicates how much data is transferred through the edge switches and it shows the importance of the network topology and application for scheduling tasks. We considered the following aspects in the experiments: (i) Effect of different mixture of requests in terms of accuracy, and (ii) Effect of different arrival rates of requests.

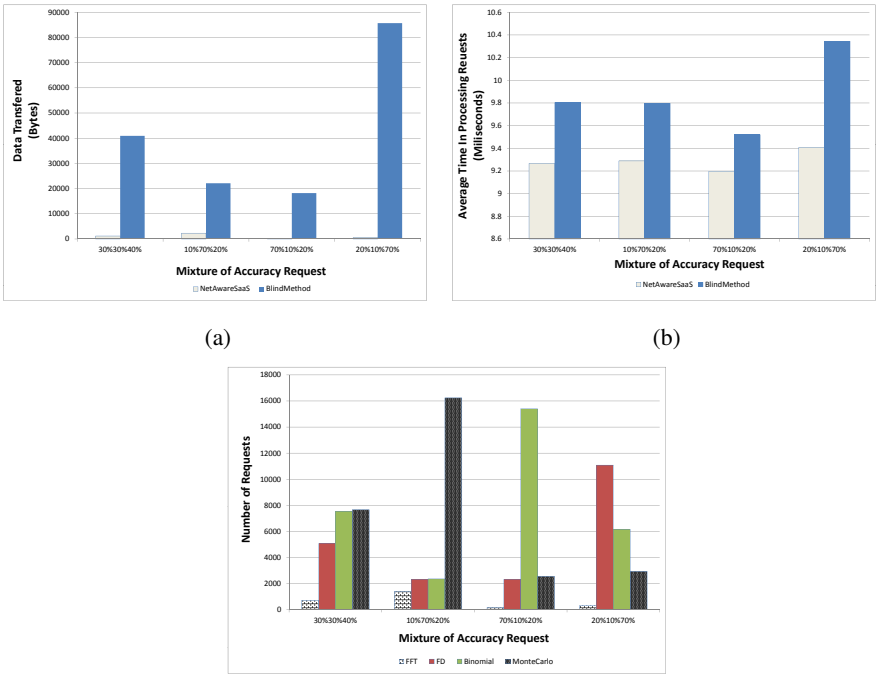

Fig. 3. Effect of Accuracy Requirements of Requests

5.4 Analysis of Results

Effect of Change in Accuracy Requirements Figure 3 presents how accuracy requirement of requests affect the performance of our proposed algorithm. Figure 3(a) , (b) and (c) present respectively data transfer, response time, and option algorithms used for our strategy (called $NetAwareSaaS$ in the figure), respectively, in comparison to $BlindMethod$, considering different rate of requests for accuracy levels. In these figures, x%,y%,z% represents the scenario where x% of requests are for low accuracy results, y% of requests are for medium accuracy results and z% of requests are for high accuracy results.

Results show that our strategy reduces network overhead due to data transfer and response time of requests, and response times and data transfers are barely affected by different mixes of accuracy, compared to the $BlindMethod$ strategy. This is due to network aware allocation of resources to serve the customer requests. Since most of the requests in BlindMethod strategy are assigned to VMs which belongs to different switches, more data communication delays resulted in more processing time. Other than this, Figure 3(c) also indicate how different ratio of accuracy requested by customers affect the choice of algorithms used for computing the option price. For instance, when 70% of requests asks for low accuracy, the large data communication is due to binomial lattice algorithm which requires about four VMs to process the customer's

request. When 70% of requests is for medium accuracy, the monte carlo algorithm is used which has low communication needs but high computation time. When 70% of request required high accuracy, the amount of data transferred across the switches is quite high in case of BlindMethod strategy due to finite-difference algorithm which has much higher data transfer requirement than any other algorithm. This also has impact on the average processing time for a request due to delays in data transfer.

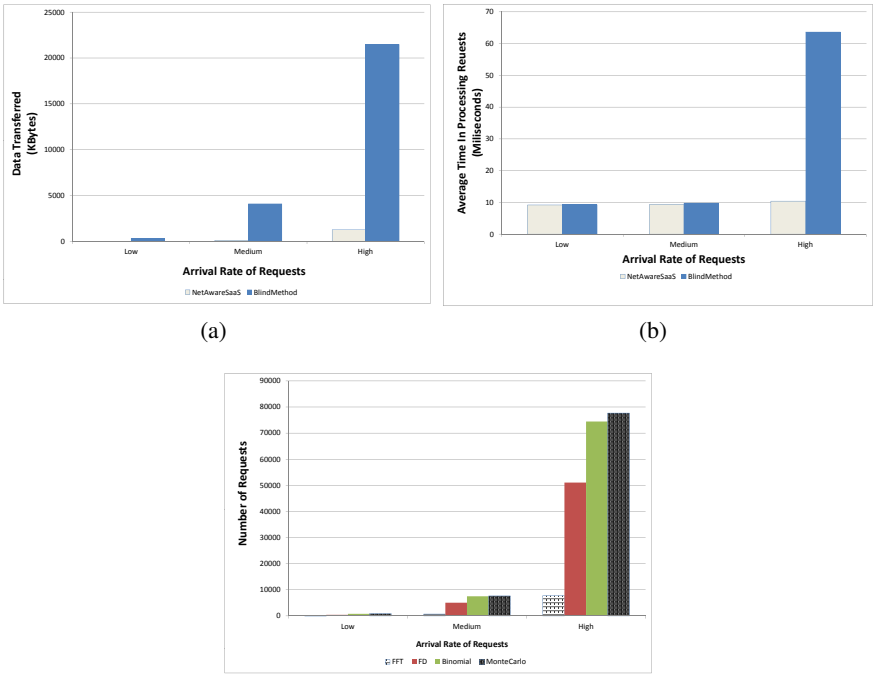

Fig. 4. Effect of Request Arrival Rate

Effect of Change in Request Arrival Rate Figure 4 presents how arrival rate of requests affect the performance of our proposed algorithm. Figure 4(a) , (b) and (c) present respectively data transfer, response time, and option algorithms used for our strategy (called $NetAwareSaaS$ in the figure) and $BlindMethod$, considering different arrival rate of requests. In the figure, low , medium, high represents the scenario where 100 requests/sec, 1000 requests/sec and 1000request/sec respectively, are received by the SaaS provider. The incoming requests have 30% low ,30% medium, 40% high accuracy requirements.

With the increase in arrival rate, the number of requests to be processed increases, and the data transferred and response time increases drastically. The reason for increase

in response time is not just the time taken to transfer data between VMs but also the queuing delay on each VM. For low and medium arrival rate the increase in response time is not as much as when arrival rate is high. This behaviour is observed due to the number of resources utilized by each strategy. For low and medium arrival rate there were enough resources to process the requests and keep the response time quite low. However, when the arrival rate increases to high, all the resources in data centers are being utilized to process each requests which also resulted in high queuing and data transferred delays. Nevertheless, the impact on the performance of NetAwareSaaS strategy is almost negligible and in all cases it leads to minimum network overhead due to data transfers and also low processing time.

6 Conclusions

In this work, we proposed a SaaS model that provides service to ordinary investors, unfamiliar with finance models, to evaluate the price of an option that is currently being traded before taking a decision to enter into a contract. The model computed as a first step, based on required accuracy and service time, an appropriate algorithm to be applied. Since these are communication intensive, we considered the communication pattern between tasks for efficient mapping to virtual machines.

Our simulation results showed that our strategy helps in reducing response time and data transfers in the internal network, compared to an approach where SaaS does not have access to the infrastructure and thus cannot apply the techniques described in this paper. Our intention is to expand the current scope of a CSP to devise algorithms for pricing such instruments as well in the near future. Portfolio optimization is another large area of service that could utilize Cloud resources.

References

1. Al-Fares, M., Loukissas, A., Vahdat, A.: A scalable, commodity data center network architecture. SIGCOMM Computer Communication Review 38(4), 63–74 (Oct 2008)
2. Barua, S., Thulasiram, R.K., Thulasiraman, P.: High performance computing for a financial application using fast Fourier transform. In: Proc. of 2005 EUROPAr)
3. Boyle, P.: Options: A Monte Carlo approach. J. of Finan. Econ. 4, 223–238 (1977)
4. Calheiros, R.N., Ranjan, R., Beloglazov, A., Rose, C.A.F.D., Buyya, R.: CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience 41(1), 23–50 (January 2011)
5. Carr, P., Madan, D.B.: Option valuation using the fast Fourier transform. The Journal of Computational Finance 2(4), 61–73 (1999)
6. Coti, C., Herault, T., Cappello, F.: Mpi applications on grids: A topology aware approach. In: Proc. of 2009 EuroPar). Delft, Netherlands (2009)
7. Cox, J.C., Ross, S.A., Rubinstein, M.: Options pricing: A simplified approach. Journal of Financial Economics 7, 229–263 (1979)
8. Hull, J.: Options, Futures, and other Derivative Securities. Prentice Hall (May 2008)
9. Kandalla, K., Subramoni, H., Vishnu, A., Panda, D.: Designing topology-aware collective communication algorithms for large scale InfiniBand clusters: Case studies with scatter and gather. In: 10th Workshop on Comm. Arch. for Clusters (CAC'10). Atlanta, USA (April 2010)

10. Lee, G., Tolia, N., Ranganathan, P., Katz, R.H.: Topology-aware resource allocation for data-intensive workloads. SIGCOMM Comp. Comm. Review 41, 120–124 (January 2011)
11. Sharma, B., Thulasiram, R.K., Thulasiraman, P.: Option pricing: A mosaic of experiments and results (Technical Memo CFD058-W10, Computational Financial Derivatives Lab, Department of Computer Science, University of Manitoba wwwcsumanitobaca/ tulsi, 2010)
12. Solomon, S., Thulasiraman, R.K., Thulasiraman, P.: Option pricing on the gpu. In: The 12th IEEE Int. Conf. on High Perf. Comp. and Comm. Melbourne, Australia (2010)
13. Thulasiram, R.K., Litov, L., Nojumi, H., Downing, C., Gao, G.: Multithreaded algorithms for pricing a class of complex options. In: Proceedings (CD-RoM) of the IEEE/ACM International Parallel and Distribued Processing Symposium (IPDPS). San Francisco, CA (2001)
14. Thulasiram, R.K., Zhen, C., Chhabra, A., Thulasiraman, P., Gumel, A.: A second order L0 stable algorithm for evaluating European options. Intl. J. of High Performance Computing and Networking (IJHPCN) 4, 311–320 (2006)
15. Volckaert, B., Thysebaert, P., Leenheer, M.D., Turck, F.D., Dhoedt, B., Demeester, P.: Network aware scheduling in grids. In: Proceedings of the 9th European Conference on Networks & Optical Communications (NOC'04). Eindhoven, Netherlands (June 2004)