INSTYTUT BADAŃ SYSTEMOWYCH
POLSKIEJ AKADEMII NAUK

TECHNIKI INFORMACYJNE TEORIA I ZASTOSOWANIA

Wybrane problemy
Tom 3 (15)

poprzednio

ANALIZA SYSTEMOWA W FINANSACH
I ZARZĄDZANIU

Pod redakcją
Andrzeja MYŚLIŃSKIEGO

Warszawa 2013

INSTYTUT BADAŃ SYSTEMOWYCH
POLSKIEJ AKADEMII NAUK

TECHNIKI INFORMACYJNE
TEORIA I ZASTOSOWANIA

Wybrane problemy
Tom 3 (15)

poprzednio

**ANALIZA SYSTEMOWA W FINANSACH
I ZARZĄDZANIU**

Pod redakcją
Andrzeja Myślińskiego

Warszawa 2013

Wykaz opiniodawców artykułów zamieszczonych
w niniejszym tomie:

Dr hab. inż. Maria GANZHA, prof. PAN

Prof. dr hab. inż. Zbigniew NAHORSKI

Dr hab. Marcin PAPRZYCKI, prof. PAN

Prof. dr hab. inż. Andrzej STRASZAK

Prof. dr hab. inż. Stanisław WALUKIEWICZ

ISBN 9788389475442

SECURITY AWARE SCHEDULING FOR PARALLEL JOB MODEL IN CLOUD SYSTEMS

Jakub Gąsior
& Franciszek Seredyński

Studia Doktoranckie IBS PAN
Department of Mathematics and Natural Sciences,
Cardinal Stefan Wyszynski University
e-mail: *j.gasior@ibspan.waw.pl*

Abstract.. This paper proposes a multi-objective parallel job scheduling algorithm for a Computational Cloud environment. We present a fault-tolerant, scalable and efficient solution for optimizing scheduling of N independent jobs on M parallel machines that minimizes two objectives simultaneously, namely the failure probability and the total completion time of all the jobs. Obtaining an optimal solution for this type of complex, large-sized problem in a reasonable computational time using traditional approaches or optimization tools is extremely difficult. As this problem is NP-hard in the strong sense, a meta-heuristic method which is the second version of the non-dominated sorting genetic algorithm (NSGA-II) is proposed to solve this problem. This approach is based on the Pareto dominance relationship, providing no single optimal solution, but a set of solutions which are not dominated by each other. The performance of the presented model and the applied GA is verified by a number of numerical experiments. The related results show the effectiveness of the proposed model and GA for small and medium-sized problems.

Keywords: multi-objective optimization; genetic algorithm; risk resilience

1 INTRODUCTION

The idea of Cloud Computing (CC), which emerged in recent years as a natural extension of Grid Computing, gradually revolutionizes a possibility of access and availability of supercomputing resources to customers without the need of possessing expensive computers, specialized software and buying licenses. It reduces the concept of computational power to a set of services, which can be bought from specialized organizations. However, after an initial enthusiastic reception of this idea by people from business and industry and getting some experience from working Cloud Computing, new issues concerning CC have been formulated. The main one can be expressed in the following way: how users can be assured that their valuable

data passed to CC system will be safe while stored, communicated and processed, and how the security of their partial or final results of computing will be guaranteed. In this paper, we aim to address such problems related to the new generation of CC, where issues of high performance and security both will be considered and new effective solutions will be worked out.

While the range of problems to be solved to provide secure computations in CC is wide, we will focus on two key issues. One of such issues is to provide a reasonable level of security-aware and robust computation in distributed systems. To obtain this, our approach proposes combination of generic methods of monitoring and discovery of anomalies and threats, leading to defining a security-assurance condition during the job mapping process for use under potentially failure-prone and risky conditions. Another key issue is providing efficient resource management of CC. An efficient resource management of parallel and distributed systems is traditionally performed by a set of load balancing and scheduling algorithms. Because load balancing and scheduling problems in a general form are known to be NP-hard, it means that a time complexity of exact algorithms delivering solutions will be at least exponential, and a provider of a system must rely on approximate solutions offered in an acceptable time using metaheuristics or heuristic approaches. One of the distinctive features of this paper is an attempt to work out a class of load balancing and scheduling algorithms which, while maintaining performance criteria, will also provide security mechanisms enabling to work in untrusted or fault-tolerant environments.

Obviously, the conflict between achieving good performance, in terms of execution time, and achieving high quality of security-assurance introduces new challenges in security critical CC scheduling. Extensive study indicates that the scheduling performance is affected by the heterogeneities of security and computational power of resources. Different jobs may have varied security requirement and even the same security requirements may exhibit different security overhead on different computational nodes. The aim of our study is to propose an efficient algorithm which effectively handles the multi-criteria parallel job scheduling problem, taking into account not only the job completion time but also the security constraints existing in a CC system.

The remainder of this paper is organized as follows. Section 2 presents the related work. In Section 3 we describe our system model. Section 4 briefs the NSGA-II algorithm and its application. Section 5 demonstrates

the performance metrics, the input parameters and experimental results. Finally, Section 6 concludes the paper.

2 RELATED WORK

Recently, a great interest of researchers in Cloud and Grid Computing domains has been focused on the secure scheduling, which aims to achieve an efficient assignment of tasks to reasonable trustful machines. The conflict between achieving good performance and high quality of security protection imposed by security-critical applications introduces a new challenge in resource allocation domain. Moreover, security heterogeneity and the dynamic property of Cloud makes solving this challenge more difficult.

A simple classification of security-aware computing models was introduced in [4], where authors defined six possible scenarios to be faced by a computing environment. An interesting approach to job-failure mechanism is presented in [5]. The authors proposed a failure detection service and a failure handling mechanism as the fault-tolerance method in Grid scheduling. It enables the detection of both task failures and user's security requirements without need of updating the protocol and the local policy at the computational nodes.

The integration of the security mechanism with the scheduling algorithms seems to be one of the most important issues in Cloud scheduling. Heuristic methods, due to their robustness, have been successfully applied to solve the large-scale task scheduling problem in the dynamic Grid environment. However, many of them cannot be directly applied in a risky environment [9]. In [1] well-known ad-hoc heuristics, namely, Min-Min, Minimum Completion Time and Sufferage methods were modified to incorporate the security requirements using a trust model for Grid systems. In [8] and [9] the authors considered the risky and insecure conditions in on-line scheduling in Grids caused by the software vulnerability and distrusted security policy. They applied a game-theoretical model introduced in [6] for modeling the resource owners selfish behavior in the hierarchical Grid structure.

The results presented in [9] were extended in [14] by considering the heterogeneity of the fault-tolerance mechanism in a security-assured Grid job scheduling. The authors defined four types of GA-based on-line schedulers for the simulation of some fault-tolerance mechanisms, including job retry, job migration (with and without check-pointing), and job replication mechanisms.

Due to the NP-hardness of the Cloud job scheduling problem, the exact solutions can not be applied to the large problems that often arise under real life scenarios. Therefore, the approximation methods that suffice to find a near optimal solution are more promising approaches. Heuristics and metaheuristics have shown to be useful approaches for solving a wide variety of hard-to-solve combinatorial and multi-objective optimization problems.

In [7] authors proposed genetic algorithm for finding a suboptimal solution to the job scheduling problem in Grid environments. In [2] genetic algorithms were used for job scheduling on computational Grids to optimize the makespan and the total flowtime. The aim of this work was to show the power of genetic algorithm in the design of effective schedulers to assign a large number of jobs originated from large scale applications to Grid resources. In [16] two implementations of cellular Memetic Algorithms (MA) were introduced for job scheduling in Grid systems when both makespan and flowtime are simultaneously minimized. MA is a relatively new class of population-based heuristic methods in which the concepts of genetic algorithm (evolutionary search) and local search are combined [15].

Our approach provides the solution for parallel job scheduling problem in distributed CC environment, while taking into account the security constraints of both user and system. We define our solution as a Pareto-based evaluation instead of more common practice of converting the multi-objective problem into a single-objective by combining the various criteria into a single scalar value or alternating them in order to optimize one criterion at a time while imposing constraints on the others. In our approach a vector containing all the objective values representing the solution fitness and the concept of dominance is used to establish preference between multiple solutions.

3 SYSTEM MODEL

In this section we formally define basic elements of the model and provide corresponding notation. Then, we define possible characteristics of the model that change the available information and the type of jobs to be scheduled. The model provides a complete representation of system resources. Additionally, it includes information about resource characteristics and availability. It provides information used by task assignment algorithms within the scheduling framework.

3.1 Definitions and Notation

The system model is an extension of the work introduced in [12] and defined as follows. A system consists of a set of m parallel machines $M_1, M_2, ..., M_m$. Each machine M_i is described by a parameter m_i, which denotes the number of identical processors P_i, called also the size of machine M_i. Figure 1(a) shows a set of parallel machines in CC system.

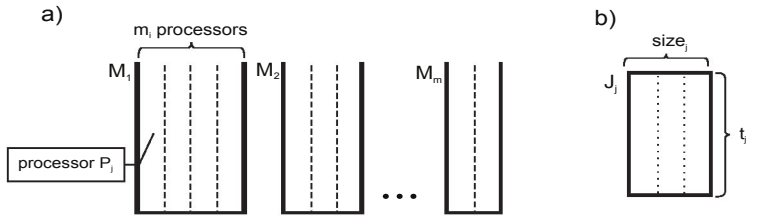


Fig. 1: Example of the Cloud Computing system. A set of parallel machines (a) and the multi-threaded job model (b).

Set of n users $(U_1, U_2, ..., U_n)$ submits jobs to the system, expecting their completion before required deadline. Thus, in the system there is a set of n jobs $J_1, J_2, ..., J_n$. A job J_j is described by a tuple $(r_j, size_j, t_j, d_j)$. The release time r_j can be defined as the earliest time when the job can be processed. In this model we assume $r_j \geq 0$. A $size_j$ is referred to as the processor requirements. It specifies a number of processors required to run the job J_j within assigned machine. We can define this as degree of parallelism or a number of threads. Rigid jobs are parallel jobs that require a fixed number of processors for parallel execution: this number is not changed until the job is completed. We assume that job J_j can only run on machine M_i if $size_j \leq m_i$ holds, that is, we do not allow multi-site execution and co-allocation of processors from different machines. Then, the $w_j = t_j * size_j$ denotes the work of job J_j. A machine executes a job of the $size_j$ when $size_j$ processors are allocated to it during time t_j, which defines required number of instructions of job J_j. Finally, d_j is the required deadline of the job. Figure 1(b) shows an example of the multi-threaded job model.

Our framework considers an open queuing network model with n machines each serving its own local queue. Arriving jobs are stored in a global queue and they are dispatched to nodes only at the end of predefined intervals called allocation intervals. We assume a fail-stop execution, meaning that if a job fails on a site, then it will be rescheduled to restart at an-

other site. Modeling the real-life scheduling situation, jobs are scheduled in batches. We assume that all jobs are scheduled independently, that is there is no communication among them. Results presented in the following sections are obtained from a single scheduling event in one allocation interval.

3.2 Security Model

We consider a security-aware scheduling scheme, to address the security issues in a computational Cloud environment. While a job is submitted in Grid, users have to define a Security Demand (SD) for that job. When setting up the SD, users should be concerned about job sensitivity, job execution environment, access control and data integration [9]. On the other hand, the defense capability of a machine is attributed to intrusion detection, firewall, anti-virus/worm, and attack response capabilities. In our paper, this defense capability is modeled by a Security Level (SL) factor. The SL evaluates the risk existing in the allocation of a submitted job to a specific machine. Each resource has its own SL value, which is one of its basic characteristics.

In our model, a job will be aborted, if the machine's SL is lower than the job SD. The SD is a real fraction in the range [0,1] with 0 representing the lowest and 1 the highest security requirement. The SL is in the same range with 0 for the most risky resource site and 1 for a risk-free or fully trusted site. Specifically, we define a job failure model as a function of the difference $SD - SL$ between the job demand and site trust. The Formula (1) presented below expresses the failure probability regarding a scheduling of a job J_j with a specific SD_j value, to the machine M_i with Security Level value SL_i [9]. The negative exponent indicates that the failure probability grows with the difference $SD_j - SL_i$:

$$P_{Failure} = \begin{cases} 0, & if SD_j \leq SL_i, \\ 1 - exp^{-(SD_j - SL_i)}, & if SD_j > SL_i. \end{cases} \quad (1)$$

3.3 Performance Metrics

Typical way of assessing system's performance is measuring the completion time of submitted jobs. There exist various levels of aggregation, on which the performance can be measured: the level of individual jobs, the level of organizations or the level of the complete system. Let us denote S

as a schedule. The completion time of jobs on machine M_i in the schedule S_i is denoted by $C_i(S_i)$. Two different objectives are considered in this work:

- The minimization of the maximum completion time C_{max}, which means the duration of all the processes. We consider minimization of the time $C_i(S_i)$ on each machine M_i over the system in such a way that the makespan is defined as $C_{max} = max_i\{C_i(S_i)\}$,
- The minimization of the total failure probability $\Sigma P_{Failure}$, defined as a sum of failure probability of each scheduling. We define the failure probability of an allocation as a function, depended on the difference between Security Demand and Security Level.

Thus, the problem can be formulated as follows: Minimize $(C_{max}, \Sigma P_{Failure})$. An important decision in such multi-objective optimization is how to evaluate the quality of solutions since the conflicting and incommensurable nature of some of the criteria makes this process more complicated.

4 GA-BASED DUAL OBJECTIVE SCHEDULING FRAMEWORK

We propose a multi-criterion Genetic Algorithm (GA) to solve the studied problem. GA is a meta-heuristic search technique that allows large solution space to be heuristically searched, by applying evolutionary techniques from nature. Typically, a GA is composed of two main components, which are problem dependent: the evaluation function and the encoding schema. The evaluation function measures the quality of a particular solution. Each solution is associated with a fitness value, which is represented by the completion time of the schedule and total failure probability. The efficiency of a GA depends on the encoding scheme applied.

4.1 Chromosome Encoding

The first step in the proposed GA is to consider a chromosome representation or solution structure. In the case of job scheduling, each chromosome represents a schedule of a group (batch) of jobs on a group of machines. A chromosome can be represented as a sequence of individual schedules (one for each machine in the group) separated by a special value. Each individual schedule is a queue of jobs assigned to that machine [11]. In this study we use the structure presented in Figure 2 for the solution of the presented model, in which each gene is a pair of values (J_n, M_m), indicating

that job J_n is assigned to machine M_m, where n is the index of the job in the batch of jobs and m is the index of the machine. The execution order of jobs allocated to the same machine is given by the positions of the corresponding genes in the chromosome on First-Come-First-Serve basis. For example, in Figure 2 Jobs 5 and 1 are allocated on Machine 1, Jobs 3 and 4 are allocated on Machine 2, while Jobs 2 and 6 are allocated on Machine 3. Algorithm scans the final solution from left to right, thus Job 4 will be executed before Job 3. This representation has been regarded in literature as efficient and compact, with reduced computational costs (crossover and mutation are easier to implement on this type of representation) [13].

Fig. 2: Chromosome encoding schema. Upper numbers define allocation sequence. A job is first scheduled to the site identified by the leftmost (J_n, M_m) pair.

4.2 Genetic Operators

Genetic algorithms produce new chromosomes by combining existing chromosomes. This operation is called *crossover*. Crossover globally searches through the solution space and produces two new offspring chromosomes (children) from the parents. Some genes of the two parents are exchanged to create the new chromosome. This operation depends on the chromosome representation, and can be very complicated. Although general crossover operations are easy to implement, building specialized crossover operations for specific problems can greatly improve the performance of the genetic algorithm. The well known standard single point crossover is adopted in this work.

Before a genetic algorithm finishes the production of a new chromosome, it performs a mutation. A *mutation* makes random, but small, changes to an encoded solution. This prevents the falling of all solutions into a local optimum and extends the search space of the algorithm. The main task of the operator is to maintain the diversity of the population in the successive generations and to exploit the solution space. In this paper, a mutation operator, called swap mutation, consists of swapping any two randomly chosen genes in a chromosome. It selects two machines, and then randomly selects a task on each machines. The tasks are interchanged between machines if they have similar properties, otherwise, the search continues.

Finally, *selection* is a process which decides about the choice of chromosomes for the crossover operation depending on the fitness function value or the rank value. In this work, the tournament parent selection is adopted. This selection is one of many selection methods in GA which randomly chooses few solutions from the parent population and selects the winner (based first on the non-dominated front, and then on the value of the crowding distance) for crossover.

4.3 Fitness Evaluation

The fitness (or objective) function measures the quality of each individual in the population according to some criteria. For the scheduling problem, the goal is to obtain task assignments that ensure minimum execution time, maximum processor utilization, a well balanced load across all processors, or a combination of these.To evaluate generated chromosomes we implement the second version of a non-dominated sorting genetic algorithm (NSGA-II) [3]. This algorithm is based on the Pareto dominance relationship. Note that in the multi-objective optimization problem, there is no single optimal solution, but a set of solutions which are not dominated by each other, as depicted in Figure 3.

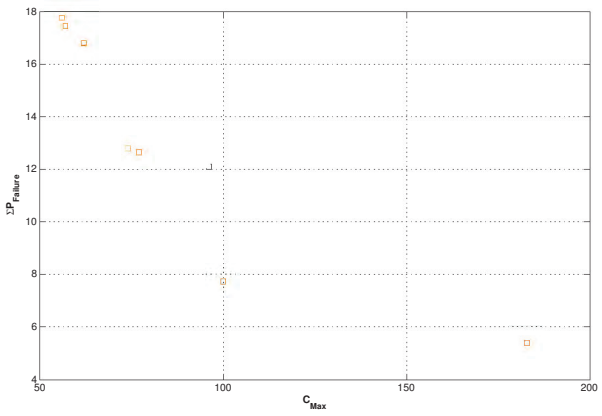


Fig. 3: An example of NSGA-II Pareto front.

In the NSGA-II algorithm, an initial population P_0 is first randomly generated. In each generation t, the following processes are executed. The population of children Q_t (all the offspring chromosomes) is created with

the operations of evaluation, selection, crossover and mutation. After that, all the individuals from P_t and Q_t are ranked in different fronts. The non-dominated front of level 1 is constituted and includes all the non dominated solutions. In order to find the solutions in the next front, the solutions of previous fronts are not considered. This process is repeated until all the solutions are ranked and assigned to several fronts. Then, the best solutions (in the best front and with the best value of the crowding distance) are chosen for the new population P_{t+1}. This generation is repeated until the stopping criterion is satisfied. The overall structure of the NSGA-II is presented in Algorithm 1.

Algorithm 1 NSGA-II Algorithm

Initialize Population P_0 of size N;
Evaluate Objective Values;
Assign Rank Based on Pareto dominance;
while *stopping criterion is not satisfied* **do**
  Generate the Offspring Population Q_t of size N (with the operations of selection, crossover and mutation);
  Compose the populations of Parents and the Offspring in $R_t = P_t \cup Q_t$ of size $2N$;
  Assign Rank Based on Pareto dominance in the combined population R_t;
  $P_{t+1} = 0$;
  $i = 1$;
  while $|P_{t+1}| + |F_i| < N$ **do**
    $P_{t+1} \leftarrow P_{t+1} \cup F_i$;
    $i = i + 1$;
  end while
  Rank the solutions of F_i by the crowding distance and
  add $N - |P_{t+1}|$ solutions in P_{t+1} by descending order of the crowding distance.
end while

There are four parameters to set in this algorithm. The population size and the number of generations define the computing time of the algorithm, the probabilities of crossover p_c and mutation p_m define the convergence and diversity of the obtained results. Table 1 summarizes those parameters.

5 PERFORMANCE EVALUATION

An important role of the scheduling process is an appropriate allocation of jobs on machines. We consider system with a different number of processors in each machine. It enforces the use of algorithms which globally distribute jobs among machines of the system. Each job should be allocated in

such a way the completion time of schedule S, as well as total failure probability is minimized. We assume that scheduler have an a priori knowledge of the service demand of each job. Thus, a matrix called Expected Time to Compute (ETC) can be generated. $ETC[i, j]$ defines the time at which machine M_i completes job J_j, after having finished any previously assigned jobs. In this section we analyze the performance of NSGA-II based scheduling algorithm for the problem defined in this paper. Its comparative performance study has been conducted with with three security-aware variants of popular scheduling policies. We briefly introduce them below.

- **Min-Min heuristic** establishes the set of minimum completion times for every unscheduled job and gives highest priority to the job that can be completed first (with minimum completion time). Selected job is then assigned to the machine which offers it this time. Min-Min executes shorter tasks in parallel whereas longer tasks follow the shorter ones;
- **Max-Min heuristic** establishes the set of minimum completion time for each job in the same way as Min-Min, but the job with the maximum minimum completion time is given highest priority in local queue and is assigned to the corresponding machine. The idea behind Max-Min is overlapping long running tasks with short running ones. Thus many shorter tasks can run in parallel with the longer ones;
- **Suffrage heuristic** is based on the idea that better scheduling results could be generated by assigning a job to a machine that would "suffer" most in terms of ETC if that particular machine is not selected. Specifically, the suffrage value of a job is the difference between its second and first earliest completion time.

All of the above heuristics operate in *f-risky mode*, that is they allocate jobs to available sites to take at most *f-risk*, where f is a probability measure defined by Equation 1. Research shows that the optimal *f-value* to achieve the minimum makespan is equal to 0.5 [10]. We adapt that value in our simulation experiments.

5.1 Input Parameters

We used in our experiments a set of randomly generated job instances. The set was generated with the following parameters: the number of jobs n_J ranging from 100 to 200 jobs, the average execution time of a job $\overline{t_J}$ was set to 5, and the average number of threads $\overline{size_J}$ was set to 4 and the average number of cores $\overline{m_i}$ was set to 6. It was assumed that the release time r was

Table 1: Simulation Parameter Settings

System Parameters	**Value setting**
Number of jobs (n_J)	100, 150, 200
Number of machines (n_M)	4, 8, 16
Average execution time of a job ($\overline{t_j}$)	5
Average number of threads ($\overline{size_j}$)	4
Average number of cores ($\overline{m_i}$)	6
Job security demand (SD)	0.6 - 0.9 uniform distribution
Machine security level (SL)	0.3 - 1.0 uniform distribution
Genetic Algorithm Parameters	**Value setting**
Number of generations	100
Population size	100
Crossover rate	0.9
Mutation rate	0.1

equal to 0 during the experiment. Deadline was not specified due to static character of the simulation. Table 1 summarizes key simulation parameters used in the experiment.

5.2 Simulation Results and Analysis

Figure 4 depicts the results of conducted experimental simulations. Batches of 100, 150 and 200 independent jobs were scheduled in the system containing 4, 8 and 16 machines. NSGA-II algorithm is based on the Pareto dominance relationship and it results in a set of solutions which are not dominated by each other. For the sake of simplicity, we present only first non-dominated Pareto front from each experimental condition containing the best overall solutions. Solutions found by three security-aware heuristics were also presented in order to provide comparative performance study of our solution.

Due to the fact that each solution is evaluated through both optimization criteria, our scheduler has the ability to predict and avoid a large amount of potential failures, while achieving reasonable level of computational overhead. Although variants of tested heuristics are security aware algorithms and give priority to higher security demanding tasks they do not achieve comparative results. It can be seen that none of the heuristics returned solution for the scheduling problem ranked in the first non-dominated Pareto front.

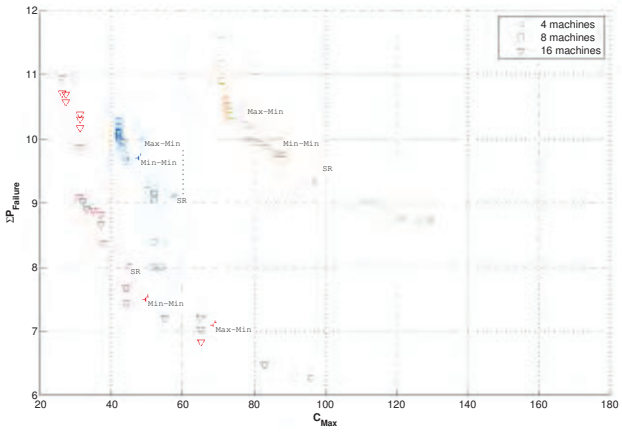
We argue that the resulting set of non-dominated solutions gives user greater flexibility and ability to prioritize his needs according to a specific criteria. Thus, it is possible to find compromise between computational overhead and desired security requirements. Our analysis shows that the application of GAs improves significantly the system's overall performance. Although GAs can cause delays to jobs until they come up with a good solution, the results strengthen the assertion that GAs are suitable for resolving large search space problems such as job scheduling in the Cloud.

6 CONCLUSION AND FUTURE WORK

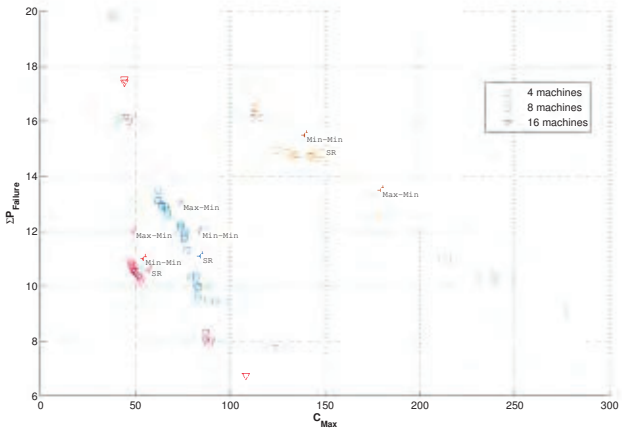
Security-driven job scheduling is crucial to achieving high performance in an Cloud computing environment. However, existing scheduling algorithms largely ignore the security induced risks involved in dispatching jobs to remote sites, which are owned by other organizations. The paper proposes a novel paradigm of GA-based dual objective scheduling for large computational cloud systems. As Cloud is considered to be a risky environment, our scheduler additionally takes into account the security constraints of the system. Thus, the proposed solution makes efforts to incorporate security into job scheduling and aims to minimize both job completion time and security risks. Non-security aware algorithms do not consider security overhead and security constraints of a job and therefore possibly assign the task to a node that only result in small computation time but a large total security overhead.

GA, due to its very nature, is capable of exploiting and exploring in the whole range of solution search space globally and picking near optimal scheduling solution. The security aware genetic algorithm makes effort to optimize quality of security and at the same time satisfy high level of performance metric. Experimental results confirm that algorithm performs better than other compared heuristics giving better completion time and security-assurance level for the same level of security.

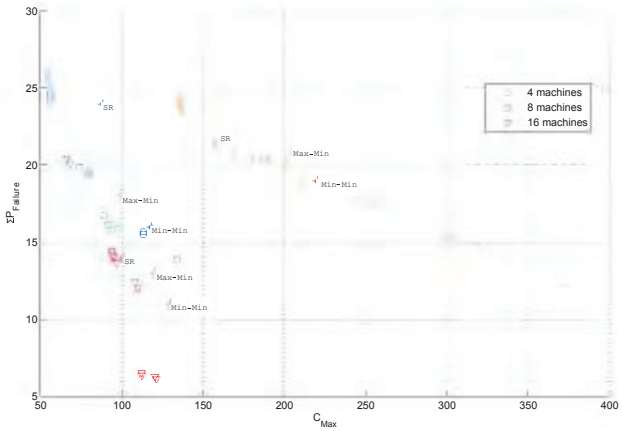
This paper could be extended by enhancing our framework with decision-making mechanisms to determine which strategies are Pareto-efficient and which of them should be applied by the scheduler without further input from the end user. We would also like to study the impact of implementing other risk-resilient scheduling strategies such as replication, check pointing and preemption on the overall performance of the Cloud system.



(a) Set of 100 jobs.



(b) Set of 150 jobs



(c) Set of 200 jobs.

Fig. 4: Results of simulation experiments employing NSGA-II Scheduler and three security-aware heuristics: Min-Min, Max-Min and Suffrage for the set of a) 100, b) 150 and c) 200 randomly generated jobs scheduled on 4, 8 and 16 machines.

ACKNOWLEDGMENT

This contribution is supported by the Foundation for Polish Science under International PhD Projects in Intelligent Computing. Project financed from The European Union within the Innovative Economy Operational Programme 2007-2013 and European Regional Development Fund (ERDF).

References

1. F. Azzedin and M. Maheswaran. Integrating trust into grid resource management systems. In *Proceedings of the 2002 International Conference on Parallel Processing*, ICPP '02, pages 47–, Washington, DC, USA, 2002. IEEE Computer Society.
2. J. Carretero and F. Xhafa. Using genetic algorithms for scheduling jobs in large scale grid applications. *Journal of Technological and Economic Development*, 12:11–17, 2006.
3. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: Nsga-ii. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pages 849–858, London, UK, UK, 2000. Springer-Verlag.
4. M. Humphrey and M. R. Thompson. Security implications of typical grid computing usage scenarios. *Cluster Computing*, 5(3):257–264, July 2002.
5. S. Hwang and C. Kesselman. A flexible framework for fault tolerance in the grid. *Journal of Grid Computing*, 1:251–272, 2003.
6. Y.-K. Kwok, K. Hwang, and S. Song. Selfish grids: Game-theoretic modeling and nas/psa benchmark evaluation. *IEEE Trans. Parallel Distrib. Syst.*, 18(5):621–636, May 2007.
7. V. D. Martino and M. Mililotti. Sub optimal scheduling in a grid using genetic algorithms. *Parallel Comput.*, 30(5-6):553–565, May 2004.
8. S. Song, K. Hwang, and Y.-K. Kwok. Trusted grid computing with security binding and trust integration. *J. Grid Comput.*, pages 53–73, 2005.
9. S. Song, K. Hwang, and Y.-K. Kwok. Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling. *IEEE Trans. Comput.*, 55(6):703–719, June 2006.
10. S. Song, Y.-K. Kwok, and K. Hwang. Security-driven heuristics and a fast genetic algorithm for trusted grid job scheduling. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers - Volume 01*, IPDPS '05, pages 65.1–, Washington, DC, USA, 2005. IEEE Computer Society.
11. R. Tavakkoli-Moghaddam, F. Taheri, M. Bazzazi, M. Izadi, and F. Sassani. Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints. *Comput. Oper. Res.*, 36(12):3224–3230, Dec. 2009.
12. A. Tchernykh, J. M. Ramírez, A. Avetisyan, N. Kuzjurin, D. Grushin, and S. Zhuk. Two level job-scheduling strategies for a computational grid. In *Proceedings of the 6th international conference on Parallel Processing and Applied Mathematics*, PPAM'05, pages 774–781, Berlin, Heidelberg, 2006. Springer-Verlag.
13. A. S. Wu, H. Yu, S. Jin, K.-C. Lin, and G. Schiavone. An incremental genetic algorithm approach to multiprocessor scheduling. *IEEE Trans. Parallel Distrib. Syst.*, 15(9):824–834, Sept. 2004.
14. C.-C. Wu and R.-Y. Sun. An integrated security-aware job scheduling strategy for large-scale computational grids. *Future Gener. Comput. Syst.*, 26(2):198–206, Feb. 2010.
15. F. Xhafa and A. Abraham. Computational models and heuristic methods for grid scheduling problems. *Future Generation Computer Systems*, 26(4):608 – 621, 2010.

16. F. Xhafa, E. Alba, B. Dorronsoro, and B. Duran. Efficient batch job scheduling in grids using cellular memetic algorithms. *Journal of Mathematical Modelling and Algorithms*, 7:217–236, 2008.

ISBN 83-894-7550-2