ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 10, Issue 3, pp: (50-61), Month: July - September 2022, Available at: www.researchpublish.com

Load Balancing with Priority Algorithm for Job Replications in Secure Computing Environments

Dlaim Alqahtani

Saudi Aramco Oil Company

Dhahran, Saudi Arabia

DOI: https://doi.org/10.5281/zenodo.7056968

Published Date: 07-September-2022

Abstract: **Distributed systems in a grid are heterogeneous resources that can be assigned to do major computing tasks using job-scheduling algorithms. Researchers have proposed job-scheduling algorithms in order to improve grid computing from different aspects such as performance, reliability, scalability, fault-tolerance, security and others. In this project, we focus on the limitations of fuzzy-logic based self-adaptive job replication scheduling (FSARS). It is appropriate for fault-tolerant and secure grid job scheduling. Moreover, it addressed the issue of fixed-number job replication scheduling and enhanced it under dynamic security conditions. FSARS can be improved in terms of performance and grid utilization by minimizing average waiting time of jobs and their average response time. Therefore, we propose a new algorithm called load balancing with priority (LBP). This algorithm is implemented for job scheduling in secure grid environment such that it gets the optimal performance compared to FSARS. LBP algorithm has two main characteristics. First, it uses load balancing between hosts that compute jobs and satisfy security conditions. Consequently, this increases grid utilization of its resources and decreases waiting time for jobs. Second, it sorts hosts based on their processing speed and their expected time to compute jobs. As a result, jobs are assigned to hosts with minimum computation time such that host does not exceed average utilization. This algorithm is implemented using MATLAB such that experimental results show improvements in terms of computing performance for job replication scheduling in secure grid environment.**

Keywords: **grid secuirty; job scheduling; grid computing; grid performance; load balancing; grid utilization.**

I. INTRODUCTION

Grids are geographically distributed systems that act together to form a virtual computing environment in order to perform large computing tasks. These tasks or jobs should be computed such that grid computing is improved from different aspects such as performance, reliability, scalability, fault-tolerance, security and others. Therefore, job scheduling is a significant part in grid computing. In addition, job scheduling has been the concentration of massive amount of research in grid computing [1] [5]. Thus, Job scheduling algorithms have a direct affect in grid computing as they are used for large-scale distributed systems.

Researchers developed grid-computing algorithms for job scheduling to achieve optimal results based on different factors to be measured in the grid. They proposed in their research several job scheduling algorithms that can lead to an efficiency of grid resources utilization. Also, they developed job-scheduling algorithms in order to get minimum computation time for jobs and obtain high level of performance in grid environment. In addition, they contributed in security enhancement for job scheduling algorithms [10]. Moreover, researchers developed algorithms such that jobs can tolerate failures by having jobs replications when they are being allocated to grid resources.

There are challenges that are found during development of job scheduling algorithms. The difficulty of job scheduling algorithm is to attain best results in all challenges. Therefore, most of algorithms focus in one challenge to be resolved but ignoring other important factors in the grid [12][13]. For example, an algorithm is designed to enhance performance and tolerate failures of the grid job scheduling but pay no attention to grid security [7]. Since grid resources are heterogonous and open, security should be maintained to protect these resources against any threat agents. Another difficulty that is hard to accomplish in job scheduling is having dynamic factors instead of static ones. Job scheduling algorithm should maintain dynamic input which is the case in real systems of grid computing. For instance, security measurements and job replications are two important factors in job scheduling algorithms. If both factors are developed as static ones, then experiment output is less realistic in case it is compared to dynamic ones [2]. Moreover, high performance is a challenge to achieve in job scheduling algorithms. This challenge can be overcome by implementing techniques that result in minimum computation time for jobs when they are assigned to hosts. Also, reducing average waiting time for jobs before they get executed can affect performance in grid scheduling algorithms. In addition, grid utilization is another challenge to define the percentage of hosts being allocated for jobs execution. Efficient grid utilization can be achieved if job-scheduling algorithm computes jobs by running minimum number of hosts in the grid sites and completing those jobs with minimum average waiting time and minimum execution time [2]. In addition, fault tolerance is a challenge in job scheduling algorithms in order to have high rate of successfully completed jobs. For example, fault tolerance can be achieved by replicating jobs when they are assigned to computing hosts in grid environment such that failures are minimized and handled during jobs scheduling [14]. Therefore, best job scheduling algorithms should address all these challenges during development process with the purpose of improving grid computation.

Many job scheduling algorithms were developed to focus on certain goals or factors in grid computation. One of the most important objectives is grid security measurement that was proposed in many research papers [1][2][5][6][8][9][11]. They focused on security as primary factor that has a direct effect on grid computing. Grid security is considered as basis to develop their grid computing algorithms. Then, researchers can add other goals to grid security goal such as algorithms that are developed for improving performance or fault tolerance but applied for secure grid computing.

Wang et al. proposed fuzzy-logic based self-adaptive job replication scheduling (FSARS) algorithm. This algorithm was developed to cover both security and fault-tolerance aspects in grid computing job scheduling [2]. Therefore, FSARS has two major factors: (1) grid security by implementing security conditions during jobs allocation to hosts, (2) fault-tolerance to minimize job failures and increase success rate of completed jobs.

FSARS algorithm used dynamicity to handle security for job scheduling. Dynamicity in security was represented in FSARS algorithm as security measurements that have different security levels based on the job security requirement. The importance of dynamic security conditions becomes clear when there is high demand for security by jobs being scheduled. In this case, this high requirement of security must be satisfied by having highly trusted hosts to be able to compute those jobs of high level of security demand. On the other hand, jobs with low level of security demand can be executed by most of hosts in the grid sites as they do not require high level security. Therefore, dynamic security conditions play an important role to satisfy different jobs security requirements [2]. Consequently, dynamic security measurement is an advantage compared to other algorithms that consider only static measurements of grid security or maintain only low level of security requirements [12][13].

FSARS algorithm also used dynamicity to handle fault tolerance for job scheduling. Fault tolerance was represented in this algorithm by job replications [2]. There are algorithms that used job replications in grid job scheduling. However, they used static and fixed number of job replications [15][16]. In FSARS algorithm, the issue of fixed-number job replication scheduling is addressed and overcome by dynamicity. In other words, dynamic job replication is used when jobs are being scheduled to hosts for computation.

Using both dynamic security measurements and dynamic job replications in FSARS algorithm has better results compared to algorithms using static factors. Results showed that jobs requiring high level security don't need high number of replications for fault tolerance. On the other hand, if jobs require low level of security, then they need a high number of replications. As a result, dynamicity in jobs scheduling algorithms for security and fault tolerance is essential in real grid environment [2].

However, FSARS algorithm did not focus on performance besides security and fault tolerance. FSARS did not consider balanced job scheduling with priority during that experiment. As a result, there are performance limitations in FSARS

algorithm that can be enhanced using either existing algorithms for performance enhancement or developing new ideas that result in improving performance metrics in job replication scheduling.

In this paper, load balancing with priority (LBP) algorithm is introduced to compute job replication scheduling for FSARS algorithm but using balanced job scheduling with priority. LBP is developed to enhance performance compared to FSARS algorithm. As in FSARS, LBP uses dynamic job replications for job scheduling under dynamic security conditions. In addition, LBP algorithm adds performance as a third factor to security and fault tolerance. Performance is enhanced in LBP algorithm using two proposed features. First, it uses load balancing between hosts that compute jobs. Second, it uses priority for jobs based on their expected time to be executed.

LBP algorithm uses load-balancing feature for job replication scheduling. In this case, average utilization is computed for all hosts in all grid sites. Hosts that are below average utilization can be used to compute jobs given that those hosts satisfy security demand of jobs being computed. Hosts that are above average utilization don't accept more jobs computation until those hosts become again below average utilization. In this case, hosts should not be overloaded and jobs are not queued in hosts for execution. Consequently, this minimizes waiting time for jobs before they executed. Otherwise, jobs wait for long time if they are assigned to full utilized host as in FSARS algorithm. Jobs in LBP algorithm are assigned to hosts that are below average utilization such that load balancing is maintained between hosts in grid sites. Thus, performance of job replication in secure grid environment is enhanced by minimizing waiting time of jobs as a result of using load balancing feature.

LBP also uses priority as another feature to enhance performance of jobs replication scheduling in grid environment. This feature is implemented by sorting hosts in grid sites based on their processing speed and their expected time to compute jobs. When jobs are submitted, they are assigned to hosts that satisfy security demand for jobs. Those trusted hosts can compute jobs if they are below average utilization to satisfy load balancing and if they also have highest processing speed among all trusted hosts set. Therefore, jobs in LBP algorithm are assigned to trusted hosts such that minimum execution time is attained. So, performance in LBP is also enhanced using priority feature.

In order to prove that LBP algorithm enhance grid performance compared to FSARS algorithm, we have to compare both algorithm results. Both algorithms are implemented by developing simulations using MATLAB. They both use same grid environment and have same configuration and setup parameters. All assumptions that were used in FSARS algorithm are also considered in our experiment so that we have fair comparison between both algorithms. LBP should show significant improvement in grid utilization and average waiting time for jobs compared to FSARS algorithm. To achieve this comparison, performance metrics are used to evaluate both algorithms after getting experiment results.

The rest of this paper is organized as follows. Section 2 is about related work which provides a summary of FSARS algorithm along with its features and limitations. Then, section 3 gives details about LBP algorithm including its architecture, design, system model, system setup, implementation, challenges. After that, section 4 presents experiment results and their analysis. Also, it includes performance evaluation of LBP algorithm compared to FSARS algorithm based on certain performance metrics. Finally, section 5 gives a brief summery and conclusion of this experiment. Also, it has future direction to extend this research for further studies.

II. RELATED WORK

Researchers have proposed several algorithms for job scheduling in grid environments. One of those proposed algorithms is FSARS [2] which is appropriate for fault-tolerant and secure grid job scheduling.

FSARS algorithm develops two main features in job scheduling in grids: dynamic job replications and dynamic security conditions. It resolved the issue of fixed-number job replication scheduling and enhanced it under dynamic security conditions. However, this algorithm was not exposed to other factors that can enhance job scheduling from performance side and increase user satisfaction with better response time. From FSARS experiment results, performance of job replication scheduling was affected by waiting time of jobs before their execution. That was caused during jobs allocation to computing hosts in grid sites such that hosts are fully utilized and jobs are not assigned to other available hosts. FSARS algorithm did not consider balanced job scheduling with priority during that experiment.

In this paper, we focus on the drawback of FSARS algorithm and develop LBP algorithm. LBP goal is to enhance grid performance compared to FSARS. Using two additional features that are added on top of dynamic security and dynamic job replication enhances Job scheduling performance in LBP algorithm. Those features are load balancing between computing

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 10, Issue 3, pp: (50-61), Month: July - September 2022, Available at: www.researchpublish.com

hosts and using priority for job scheduling by sorting computing hosts by their processing speed and their expected time to compute jobs. There are algorithms that proposed different load balancing algorithms in grid computing using different policies [3][4]. However, LBP uses different load balancing methodology.

LBP algorithm is developed to get optimal results in three areas: fault tolerance, security, and performance. Load balancing between hosts in grid sites can decrease average waiting time for jobs during their scheduling. Also, efficient use of grid resources is attained by load balancing technique in LBP because jobs are directed only to hosts which are available and trusted. Therefore, busy hosts are always avoided during jobs allocation which provides an advantage of LBP over FSARS. In addition, priority for jobs scheduling based on hosts processing speed can decrease the total time for jobs execution because jobs are assigned to hosts that have high processing speed and have minimum expected time to process jobs. As a result, jobs are allocated to available and trusted hosts that can process them faster. In addition, LBP uses dynamic job replications with dynamic security conditions.

III. THE LBP ALGORITHM

In LBP algorithm, a set of jobs are assigned to a set of hosts in grid sites. In this paper, jobs and tasks are used to replace and represent each other. The set of jobs are represented in LBP by $T = \{T_1, \ldots, T_i \ldots, T_n\}$. The set of computing hosts are represented by $M = \{M_1, \ldots, M_j, \ldots, M_m\}$. Each job can be represented by $T_i = (SD_i, K_i, H_i, Q_i, SE_i)$, where SD_i is security demand required for the job T_i to be successfully executed, K_i is the job replication number, H_i is the total heat of the job, Q_i is the set of hosts that are trusted by the job T_i, and SE_i is security error percentage for the job T_i. Each computing host is represented by $M_j = (TL_i, P_j, GTL_j)$, where TL_i is trust level of host M_j, P_j is processing speed of the host, and GTL_j is trust level of the gird. In addition, E_{ij} is the expected time for the host M_j to compute the job T_i.

Jobs scheduling in LBP algorithm use security demand SD_i as dynamic such that each job has its own security demand independently from other jobs. SD_i is assigned to each job once submitted for scheduling. SD_i can take values between 0 and 1 such that small values of SD_i represent low security demand and big values represent high security demand. K_i is also dynamic such that each job is assigned a dynamic replication number. Jobs that have low values of security demand require high number of replication in order to be completed successfully. K_i takes values from 1 to 4 in LBP to mimic FSARS algorithm setup parameters but dynamically generated for every submitted job [2]. Heat of a job is calculated by product of its expected completion time and job replication number and can be represented by:

$$H_i = K_i * E_{ij} \qquad (1)$$

Job replication scheduling in LBP uses security conditions to have a successful execution of jobs. When job T_i is submitted and it has particular security demand SD_i, then following security condition must be true in order to determine set of trusted hosts of that individual job:

$$SD_i \leq TL_j \qquad (2)$$

Q_i represents the set of trusted hosts that satisfy this security condition in (2) for a given job T_i. Hosts that belong to Q_i are the only hosts in grid sites that compute the job T_i. Other hosts that are not in the set of Q_i cannot execute that job as they don't satisfy (2). Q_i is different between from one job to another because each job has different security demand. As a result, set of trusted hosts Q_i will change accordingly. After that, Q_i is used during process of load balancing with priority such that this process is applied only for the Q_i set and the job can be directed only to hosts in the Q_i set.

Each job has a security error ratio such that it provides the difference between the security demand of the job and the trust level of the whole grid site. SEi can have values between -1 and $+\infty$ as in the case of FSARS algorithm [2].Security error ratio of the job T_i is calculated by:

$$SE_i = (GTL_j - SD_i) / SD_i \qquad (3)$$

Since grid computing uses heterogeneous resources to execute jobs, each host in the grid sites has its own processing speed P_j. In addition, each host has a trust level TL_j such that it cannot execute a given job unless it is trusted by that job. Each host computes the job and we can get the time taken for a host M_j to complete the job using E_{ij}. Then, total time to complete jobs in LBP can be computed by calculating the sum of E_{ij} for all jobs. Each time taken by host to compute the job is multiplied by the job replication number to get the job heat. Therefore, we can use job heat to compute grid average load.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 10, Issue 3, pp: (50-61), Month: July - September 2022, Available at: www.researchpublish.com

LBP algorithm (see Fig. 1) is developed to enhance grid performance compared to FSARS algorithm. LBP algorithm uses job replication and security measurements that were done in FSARS. In addition, LBP algorithm adds two features: (1) load balancing between hosts; and (2) priority for jobs based on their expected time to be computed. The algorithm maintains load balancing by computing average utilization of hosts in the grid (see steps 1-7). Average utilization is calculated using total time (H_{sum}) to compute all jobs (T_i) and their replications (K_i) divided by number of hosts in the grid. After that, algorithms apply security measurements to the hosts in the grid (see steps 9-13). To compute a job successfully, its security demand (SD_i) should not exceed the trust level of the host (TL_j). Otherwise, the job cannot be computed due to security risk in the grid (see steps 23-24). If there is a set of hosts that satisfy this security condition for that job, then this set (Q_i) is sorted based on expected time to compute the job by a given host (see steps 14-17). Finally, average utilization computed earlier is used in steps 18-22 to maintain load balancing between hosts in the grid. The job is assigned to the host that satisfies security conditions, computes the job with minimum time period, and does not exceed grid average utilization.

```
1.   Compute the average grid utilization
2.   Hsum = 0
3.   for each job Ti do
4.       Hi = Ki * Eij
5.       Hsum = Hsum + Hi
6.   end for
7.   Avg_Util = Hsum/Host_Num
8.   for each job Ti do
9.       for each host Mj in the grid do
10.          if ( SDi ≤ TLj )
11.          then Mj ∈ Qi
12.          else  Mj ∉ Qi
13.      end for
14.      for each host Mj in Qi do
15.          Compute Eij
16.      end for
17.      Sort hosts Mj in Qi based on Eij
18.      for each host Mj in Qi (sorted) do
19.          if ( loadj ≤ Avg_Util )
20.          then Assign Ti to Mj
21.          else check next Mj in sorted list
22.      end for
23.      if (Qi is empty)
24.      then job failed
25.  end for
```

Fig 1:The LBP Algorithm

Fig. 2 shows a logical system diagram of how LBP algorithm works. It starts with a defined set of jobs that can be submitted for computation by hosts in grid sites. Each job should use adaptive replication number (K_i) and has its own security demand (SD_i). Then, the job security demand is compared against all hosts trust level. Therefore, we get one set of trusted hosts and another set of non-trusted hosts for every submitted job. The membership of a computing host to one of the sets is based on the comparison of the condition in Fig. 2 between job security demand and the host trust level. After that, the job enters the state where it has only a set of trusted hosts such that the job passed the security condition. Next phase is the major part of LBP algorithm where we have the two main phases: load balancing and priority.

The hosts are prioritized based on their processing speed in all grid sites. Hosts with high processing speeds have higher priorities than hosts with low processing speed. The job is assigned to the host with highest priority if that host load does not exceed the average load of all hosts. The average load of all hosts is updated with every job execution by any host. If that host load is greater than the average load at the time of job assignment, then the job will check for next host in the priority

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 10, Issue 3, pp: (50-61), Month: July - September 2022, Available at: www.researchpublish.com

list and so on until it finds the host whose load is less than grid average load. Once that host is found, it executes the job so that we get expected time of job completion. Using this priority technique in LBP algorithm helps the job to be assigned to the host to get minimum execution time. In addition, using load balancing methodology helps to assign jobs to available hosts without having to wait longer in jobs queue. As a result, the job execution by a particular host is completed and execution time is recorded for that job.

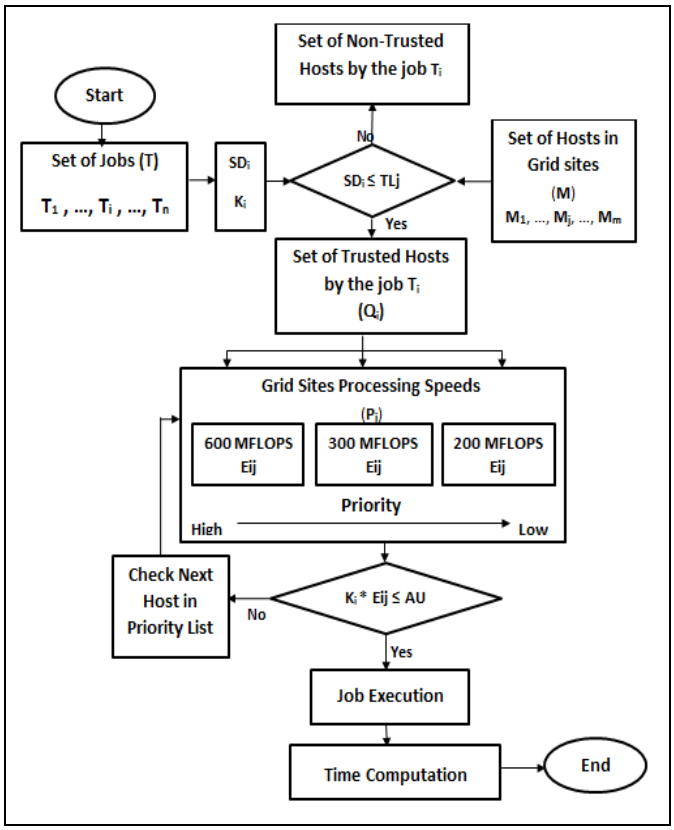

Fig 2: System Model

FSARS simulation is implemented using MATLAB to be able to compare it with LBP algorithm. We have developed MATLAB code to simulate FSARS algorithm. The FSARS simulation shows that we got the expected results which match the results in FSARS algorithm paper [2]. The simulation contains our implementation code of all functions required for fault tolerance and security measurements. This simulation does job scheduling using adaptive replication number. Then, there is a function using MATLAB to do random assignments of those job replications to hosts. If at least one job replication found a host that satisfies the security requirement of that job, then the job has no security error. However, if all job replications were directed to hosts which do not satisfy job security demand, then job security error occurs with no assignment to any host. Therefore, the job has to be resubmitted again along with its replications such that it may find a host that satisfy security requirement. This security error is also being calculated inside the FSARS simulation. After that, all jobs that are assigned to trusted hosts can be computed. We developed the simulation code such that it can calculate the waiting time and execution time of every job which is assigned to a trusted host. In addition, we measure the efficiency of grid resources utilization in FSARS simulation by calculating how many hosts are used to compute the jobs. Finally, we take all calculated results from FSARS simulation to compare it with LBP algorithm.

LBP algorithm is developed also using MATLAB. All functions required to implement LBP algorithm are developed to be able to compare the results against FSARS algorithm. MATLAB code for LBP starts with loading all configuration parameters that are used for FSARS. Then, it checks for security demand of the job and compare it with trust level of all grid hosts. Therefore, we get to sets of hosts: trusted hosts and non-trusted hosts. After that, the job replication is assigned to the set of trusted hosts. By this technique, we guarantee that the job replications are assigned only to hosts that are trusted. Thus, this is an advantage of LBP algorithm over FSARS because jobs using FSARS assign the job replications to hosts before they check their trust level. Then, if all hosts are not trusted, FSARS jobs will fail with security error. However, jobs of LBP algorithm check first the security measurements and isolate non-trusted hosts for a given job. So, LBP jobs should not have security errors compared to FSARS.

MATLAB code of LBP algorithm also implements the technique of load balancing with priority. After the jobs are directed to the set of trusted hosts, the code implements the priority methodology of those trusted hosts based on their processing speed. Only trusted hosts of a given job are prioritized for that job. So, every job has its own list of trusted hosts that are prioritized based on their processing speed and expected time of completion. Hosts having high processing speed have higher priorities than hosts having slow processing speeds. The job checks the load of the first trusted host in its priority list. If its load is less than current average load of all hosts, then the job will be assigned to this host for execution. However, if the load of that trusted host is greater than current average load of all hosts, then job check next host in its priority list. The job keeps going over its priority list until it is assigned to the host whose load is less than average load of all hosts.

Load balancing is maintained in MATLAB code by checking the average load of all hosts for every job and for every trusted host in priority list of every job. Load balancing technique is developed to avoid overload hosts and also minimize waiting time of the job if the host is busy. As a result, the job is always assigned to the trusted host that can execute the job in minimum time.

The LBP algorithm functions are implemented using MATLAB to obtain required results. There is a function to record all job executions to get the total time of jobs execution. Also, another function is developed to compute the waiting time for every job and then compute the average waiting time for all jobs. Based on LBP techniques, both average waiting time and execution time should be improved. Also, we developed our code to calculate the efficiency of resource utilization. That is computed using number of hosts that are used to execute jobs out of the total number of hosts. LBP should show that less number of hosts is used to execute jobs while FSARS use more number of hosts to execute the same number of jobs.

At the end of implementation of LBP using MATLAB, we develop functions that can show comparison between all functions included in both FSARS and LBP algorithm. The comparison is done to show the performance differences between both algorithms in following areas: total execution time of jobs, waiting time for jobs, and efficiency of grid resources utilization. In addition, the comparison covers the security side of both algorithms. MATLAB code is implemented to include security error ratio comparison. The comparison is to show how many jobs in each algorithm can fail because of unsatisfied security measurements.

Finally, we went through several technical challenges to develop LBP algorithm and in order to show that there is a significant enhancement based on load balancing and priority techniques. Also, LBP is designed to show its superiority in performance compared to FSARS. One of the challenges is to simulate FSARS algorithm because it was implemented by using a computing application which is designed for testing grid applications [2] and we had to develop this algorithm using MATLAB with our own implementation code. In addition, another difficulty is to show that all results in our code are the same as the results of FSARS paper. Another challenge is to implement LBP using the same configuration setup which was used by FSARS in order to get better results. The design for LBP had major two stages such that each phase should show an enhancement in the final outcome of our experiment. In the first stage, difficulties arise when we need to minimize security error percentage in FSARS which was 20%. That was an area of improvement and we found an idea of separating trusted hosts from non-trusted hosts that can help to minimize this security error ratio. In the second stage, the difficulty is to come up with the idea of improving three main factors: jobs execution time, jobs average waiting time, and efficiency of grid utilization. Joining both new techniques load balancing between all gird hosts and using priority based on grid sites processing speeds was the idea to achieve this algorithm. After overcoming all these technical challenges, LBP shows excellent results in performance and security aspects as explained in details in following section.

IV. PERFOMRANCE EVALUATION

In this section, we evaluate the performance of both FSARS and LBP algorithms based on our experiment outcome. Initially we present the configuration setup for both algorithms including system parameters and basic elements of the experiment. Also, we show the factors that we used to measure performance of both algorithms. Then, we show the results obtained from our experiment by using MATLAB. Based on these results, both algorithms are compared in terms of performance. We have performance metrics that are used in this comparison. Results will show that LBP algorithm has significant improvement in performance when it is compared to FSARS algorithm performance.

A. Experimental Environment Setup

The LBP algorithm is implemented to improve results of experiments that were performed in FSARS algorithm. In this project, we use a set of hosts that compute a set of jobs in grid sites by developing simulations in MATLAB. Network communications are not considered as a factor that can have any effects on this experiment. Also, jobs don't have deadline such that we can measure average waiting time for jobs scheduling.

Following are key components in our experiments for LBP algorithm:

- Grid hosts which compute jobs and maintain load balancing based on expected jobs execution time.
- Jobs are secured during their assignments to hosts by using jobs security demand and hosts trust level.
- Security measurements are validated during process of job scheduling.
- Job replication is dynamic to produce fault-tolerance job scheduling.
- Grid hosts are prioritized based on their processing speed and expected time to complete jobs.
- Grid hosts maintain load balancing during job replication scheduling.

The experiment has a set of parameters that were used also in FSARS algorithm. Table I shows this list of system parameters to build simulated secure and fault-tolerant grid environment of job replication scheduling. We have set these parameters for both FSARS and LBP algorithms in MATLAB simulation to perform this experiment.

TABLE 1: SYSYEM PARAMETERS

Parameter	Value
Number of jobs	20000
Number of grid sites	128
Speed of hosts (MFlops)	200,300,600
Job security demand	(0-1)
Expected computation time (s)	(120-400)
Trust level of hosts	(0.2-1)
Replication number per job	(1,2,3,4)

In order to evaluate LBP algorithm, we have following metrics for performance evaluation:

- Increase efficiency of grid resources utilization when using LBP algorithm.
- Minimize the total execution time that jobs take to complete their runs.
- Decrease average waiting time of jobs by sorting hosts that satisfy security conditions based on expected time to compute those jobs.
- Minimize job failures that may occur because of security conditions or overloading grid sites by separating trusted hosts from non-trusted hosts for every submitted job.

Compared to FSARS algorithm, the LBP algorithm should show in next section improvements in terms of those four performance metrics in our experiment.

B. Experimental Results

Since the experiment of LBP algorithm uses load balancing between hosts that satisfy security measurements, this increases efficiency of grid resources utilization In addition, it decreases waiting time for jobs before they get executed. The absence of this feature in FSARS algorithm results in some hosts which are available and have high processing speed to be low utilized. Consequently, FSARS can decrease efficiency of grid resources utilization. In addition, jobs that are scheduled using FSARS can wait longer if they are assigned to over utilized hosts. Therefore, experiment of LBP algorithm uses load balancing to satisfy these two performance metrics: maximum grid utilization and minimum average waiting time for job replication scheduling.

Another major improvement in the experiment of LBP algorithm is to sort hosts that satisfy security conditions for a given a job. We sorted hosts in MATLAB code based on expected time to execute a given a job. As per FSARS algorithm, we assume that expected time to compute a job in any host in the grid sites is known. Since hosts have different speeds in the grid, their computation times for jobs also differ. As a result, the experiment assigns the job for the host with minimum computation time such that host does not exceed average utilization.

In this experiment, we observed that having load balancing and sorting hosts based on computation time can enhance performance metrics using MATLAB. MATLAB code simulates both FSARS and LBP algorithms such that both run under same environment and same configuration parameters. Also, all assumptions during FSARS experiment were considered in our experiment to be able to fairly compare the results between the two algorithms. Experimental results show that there is significant improvement using LBP over FSARS in terms of performance and security areas. From performance side, it shows improvement in total execution time of jobs, average waiting time of jobs, and efficiency of using hosts in the grid sites. From security side, it shows improvements in the security error ratio for jobs replication scheduling using LBP.

Total execution time of jobs shows superior results in LBP algorithm when it is compared to the results of FSARS algorithm as displayed in Fig. 3. In this graph, we show a set of 20 jobs in order to show clearly the difference between the two algorithms. Also, we have used in our experiment the maximum number of jobs that is set to 20,000 as per the system parameters. Also, we used the number of gird sites to be 128 sites. After setting all the system parameters as explained in Table I, we have computed using MATLAB code the total execution time of 20,000 for both LBP and FSARS algorithm. Then, we computed using our code the average execution time of all those jobs. As a result, we found that based on MATLAB results, the average execution time per job using LBP algorithm is 130 seconds while the average execution time per job using FSARS algorithm is 200 seconds as shown in Table II. In other words, the average execution time per job using LBP algorithm is decreased by 35% when it is compared to average execution time per job FSARS algorithm. That is because we used priority technique, which helps the job to be assigned to the host in the priority list such, that job is executed with minimum execution time. Therefore, LBP provides much better job execution time compared to FSARS.

Average waiting time also shows better results in LBP algorithm versus FSARS algorithm as shown in Fig. 4. In this graph, we show the set of 100 jobs in order to make it clear for displaying the difference between results of both LBP and FSARS algorithms. In our experiment, we also used the maximum number of jobs which is 20,000 as set by the system parameters. The results based on MATLAB shows that there is an enhancement in the average waiting time. The waiting time is computed for every job assignment in both algorithms. Then, the average waiting time is computed for all 20,000 jobs using our code. The output shows that average waiting time of jobs using LBP algorithm is 13 milliseconds. However, the average waiting time of jobs using FSARS is 30 milliseconds as shown in Table II. Consequently, average waiting time of jobs replication scheduling using LBP is decreased by 56.6% by comparing it to the average waiting time of jobs using FSARS algorithm. This is a result of using load balancing between hosts such that they do not exceed average load. Then, jobs waiting time is minimized during their scheduling. Thus, these results show that LBP perform better than FSARS algorithm in terms of jobs waiting time.

Security error ratio shows also good results when we compare LBP to FSARS as illustrated in Fig. 5. We used also a set of 20 jobs to show clearly the difference between both algorithms. In this graph, we show using MATLAB the difference of security error ratio in both algorithms. Equation (3) is applied in MATLAB to compute the security error ratio. In our experiment, we used number of 20,000 jobs to see the results in both algorithms. Experiment results shows that 10% of the jobs failed to find a trusted host. On the other hand, 0% of jobs have security error using LBP algorithm. Thus, job security error is decreased by 10% when using LBP algorithm as shown in Table II. This enhancement is due to the separation of trusted hosts from non-trusted hosts such that the job replication scheduling is directed only to set of trusted hosts. Therefore, we see from those results an enhancement of LBP algorithm in terms of performance and security for job replications scheduling. This is shown clearly in the results of comparing LBP and FSARS algorithm.

In addition, efficiency of grid resources utilization is computed using MATLAB as shown in Fig. 6. The results show that LBP uses gird resources more efficiently than FSARS. In our experiment, we used 20,000 jobs to be scheduled and assigned to 128 hosts in different grid sites. In FSARS algorithm, all 128 hosts were used to compute the 20,000 jobs. As a result, percentage of hosts that were used to compute jobs out of the total hosts is 100%. However, only 111 hosts were used to compute the 20,000 jobs. Therefore, only 86.7 % out of the total hosts were used to compute all the jobs as displayed in Table II. This improvement is due to the usage of load balancing method along with priority technique because hosts are reused once their loads become less than average load which is dynamically updated with every job assignment. In addition, hosts with least priority have less probability to compute jobs.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 10, Issue 3, pp: (50-61), Month: July - September 2022, Available at: www.researchpublish.com

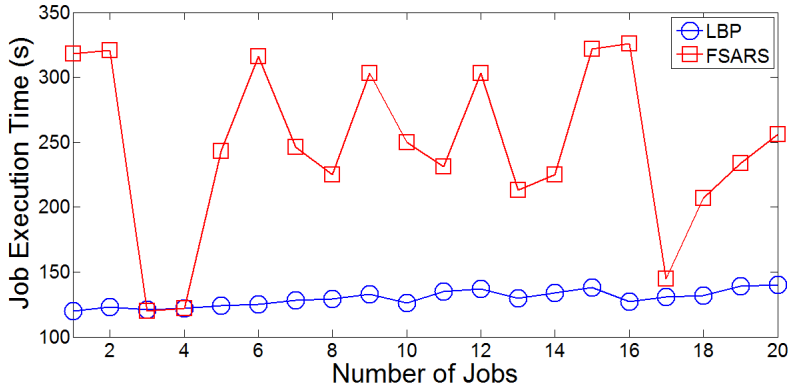

Fig 3: Jobs Execution Time in LBP and FSARS Algorithms

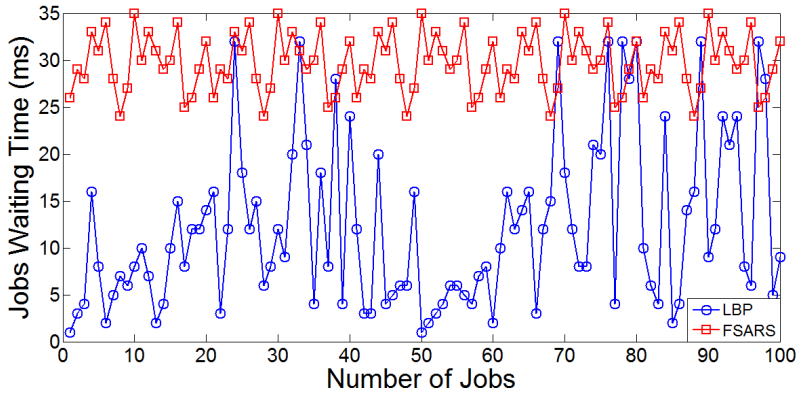

Fig 4: Jobs Waiting Time in LBP and FSARS Algorithms

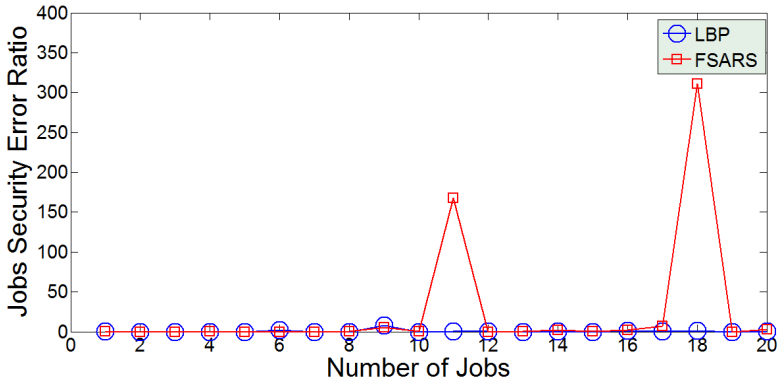

Fig 5: Jobs Security Error Ratio in LBP and FSARS Algorithms

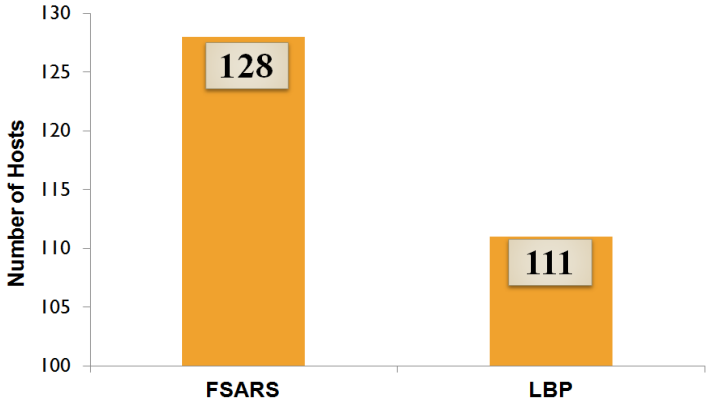

Fig 6: Grid Resources Utilization in LBP and FSARS Algorithms

TABLE 2: EVALUTATION

Performance Metrics	FSARS	LBP
Makespan: jobs average execution time	200s	130s
Average waiting time of jobs	30ms	13ms
Percentage of jobs with security error	10%	0%
Used hosts out of total hosts	100%	86.7%

V. CONCLUSION

In this paper, we developed LBP algorithm for job replication scheduling in secure grid environments. We focused on FSARS algorithm [2] that is applicable for fault tolerance and secure job replication scheduling. We analyzed also several algorithms that discuss load balancing [3][4] in secure grid computing environments. We developed our own LBP algorithm and implemented it using MATLAB by performing several experiments in order to find optimal performance in job replication scheduling. This algorithm has been designed in three major phases. First phase implements dynamic jobs replication scheduling to provide fault tolerance for grid computing. Also, this phase includes dynamic security measurements that are used to determine trusted hosts from non-trusted hosts for a given job. Second phase implements the priority technique such that computing hosts are sorted by their processing speeds in the grid sites. The priority increases as the processing speed increases. Third phase implements the load balancing method by computing the average load of all hosts. Then, all trusted hosts load should not exceed the average load of all hosts so that they can execute jobs. Otherwise, next trusted host in the priority list is check with average load again to be scheduled for job execution.

Experimental results of LBP show significant improvement compared to the results of FSARS algorithm. Using MATLAB, the results of our experiment of LBP shows that average execution time is decreased by 35% compared to FSARS algorithm because of using priority technique. In addition, average waiting time of job replications scheduling is decreased by 56.6% compared to average waiting of jobs using FSARS. In addition, security error ratio is decreased by 10% when using LBP versus FSARS. Security error ration represents the number of jobs that failed due to unsatisfied security conditions. Moreover, efficiency of resource utilization resulted in using only 86.7% of the hosts to compute 20,000 jobs in LBP while 100% of the hosts were used to compute the same number of jobs in FSARS algorithm.

Future studies in this research can be performed in the following directions. First, enhance job replications such that performance should not be affected if job replication number increases. In FSARS, they could not maintain good performance when the dynamic job replication number is more than 4. This was not in our scope of work in this paper as we focused on performance and security enhancement.

In addition, fault-tolerance in current algorithms for secure grid computing can be improved by developing algorithms that can minimize points of failures in grid computing algorithms. This can improve resources availability and maintain their communication during job assignments.

Simulating these proposed algorithms in real systems or computing applications could do other future experiments. Therefore, running multiple tests on these systems can test scalability and other factors.

REFERENCES

[1] T. Xie, and X. Qin, “SAHA: A scheduling algorithm for security-sensitive jobs on data grids,” CCGrid, pp. 22, 2006.

[2] W. Cheng, J. Congfeng, L. Xiaohu, “Fuzzy logic-based secure and fault tolerant job scheduling in grid,” Journal of TSINGHUA Science and Technology, Vol. 12, No. S1, pp. 45-50, July 2007.

[3] K. Lu, A.Y. Zomaya, “A hybrid policy for job scheduling and load balancing in heterogeneous computational grids,” ISPDC, pp. 121-128, 2007.

[4] J. Ma, “A novel heuristic genetic load balancing algorithm in grid computing,” IHMSC, pp. 166-169, 2010.

[5] H. Zhu, Y. Wang, Z. Ma, H. Li, “Grid dependent tasks security scheduling model and DPSO algorithm,” Journal of Networks, Vol. 6, No. 6, pp. 850-857, June 2011.

[6] O. Linda, M. Manic, T. Vollmer, “Improving cyber-security of smart grid systems via anomaly detection and linguistic domain knowledge,” ISRCS, pp. 48-54, August 2012.

[7] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman, "Heuristics for scheduling parameter sweep applications in grid environments," HCW, pp. 349-363, May 2000.

[8] S. Song, K. Hwang, Y.K. Kwok, "Trusted grid computing with security binding and trust integration," Journal of Grid Computing, Vol. 3, pp. 53-73, June 2005.

[9] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, "A security architecture for computational grids," 5th ACM conference on computer and communications security, pp. 83-92, 1998.

[10] R. Fotohi, M. Effatparvar, "A cluster based job scheduing algorithm for grid computing," IJITCS, pp. 70-77, November 2013.

[11] T.V. Ryutov, B.C. Neuman, "The specification and enforcement of advanced security policies," Conference on Policies for Distributed Systems and Networks, pp. 128-138, June 2002.

[12] T.D. Braun, D. Hensgen, R. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," Journal of Parallel and Distributed Computing, Vol. 61, pp. 810-837, June 2001.

[13] A. Dogana, F. Ozguner, "Scheduling of a meta-task with QoS requirements in heterogeneous computing systems," Journal of Parallel and Distributed Computing, Vol. 66, pp. 181-196, February 2006.

[14] S. Hwang, C. Kesselman, "A flexible framework for fault tolerance in the grid," Journal of Grid Computing, Vol. 1, No. 3, pp. 251–272, 2003.

[15] S. Song, K. Hwang, Y. Kwok, "Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling," IEEE Transactions on Computers, Vol. 55, No. 6, pp. 703-719, 2006.

[16] J.H. Abawajy, "Fault-tolerant scheduling policy for grid computing systems," IPDPS, April 2004.