International Journal of Computer Science and Information Technology Research (IJCSITR)
Vol. 1, Issue 1, pp: (42-50), Month: October-December 2013, Available at: www.researchpublish.com

Design and Analysis of Security Aware Scheduling in Grid Computing Environment

Mudassir Khan

Department of Computer Science
King Khalid University, KSA

***Abstract*-- Grid computing solves large scale applications by coordinating and sharing computational power, data storage and network resources across dynamic and geographically dispersed organizations by providing high performance computing platform with the goal of providing users with access to the resource they need, even when they need. Grids provide remote access to IT assets, and aggregate processing power. The goal of scheduling is to achieve highest possible system throughput and to match the application need with the available computing resources. Scheduling on to the Grid is NP complete, so there is no best scheduling algorithm for all grid computing systems. The basic grid model generally composed of a number of hosts, each composed of several computational resources, which may be homogeneous or heterogeneous. In recognition that many applications running on Grids demand both real-time performance and security. In this research proposal the problem of Security Aware- Scheduling on real time applications with various security requirements for Grid Computing environment is discussed. In the work security aware scheduling models with better load distribution for achieving a flexible trade-off between overhead caused by security services and system performance are proposed considering security middleware model (SMW) and its peer technologies by which security-sensitive real-time applications are enable to exploit a variety of security services to enhance trustworthy executions of the applications.**

Keywords- Grid Computing, Grid Scheduling, Security aware model

1. Introduction

The growing number of devices and thus mostly unused resources connected to the internet triggered many different ideas to share available computing and storage resources. In 1998 Ian Foster and Carl Kesselman defined in the book The Grid: Blueprint for a new Computing Infrastructure a computational grid as a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexperience access to high-end computational capabilities. In general, grid computing provides users with the ability to divide and spread large computations across multiple machines as well as access to distributed storage and collaboration possibilities within virtual organizations. Grid allows the simultaneous use of large numbers of resources, dynamic requirements, use of resources from multiple administrative domains, complex communication structures, and stringent performance requirements.

The goal of a grid computing, like that of the electrical grid, is to provide users with the access to the resources they need, when they need them. Scheduling on to the grid is NP complete, so there is no best scheduling algorithm for all grid computing systems. An alternative is to select an appropriate scheduling algorithm to use in a given grid environment because of the characteristics of the tasks, machines and network connectivity. Security Aware Scheduling is one of the key research areas in grid computing. The goal of scheduling is to achieve highest possible system throughput and to match the application need with the available computing resources [1-3].

2. Basic Grid Model

The basic grid model generally composed of a number of hosts, each composed of several computational resources, which may be homogeneous or heterogeneous. The four basic building blocks of grid model are user, resource broker, grid information service (GIS) and lastly resources [3].

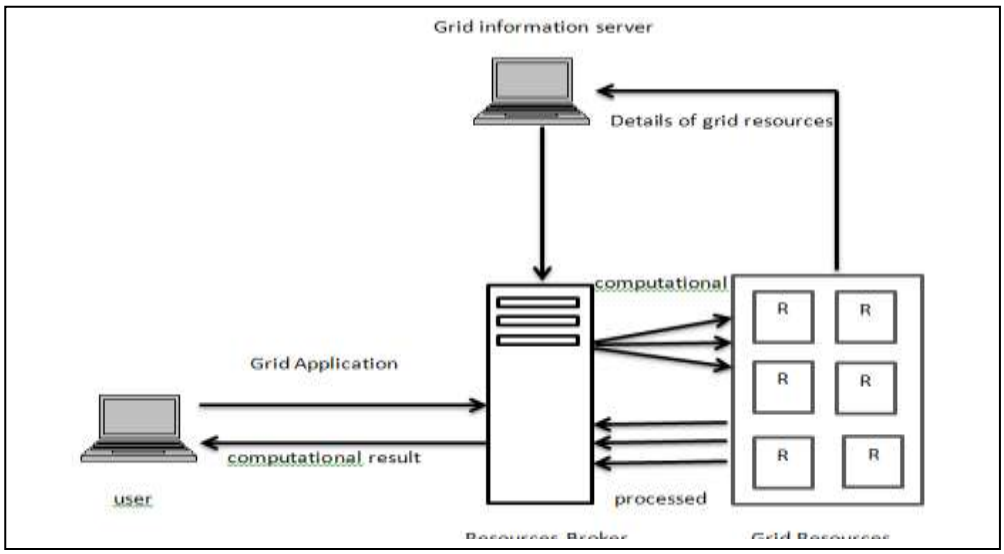

Figure 1: Basic Grid Model [3]

When user requires high speed execution, the job is submitted to the broker in grid. Broker splits the job into various tasks and distributes to several resources according to the user requirements and availability of resources. GIS keeps the status information of all resources which help the broker for scheduling.

Security aware scheduling provides an efficient selection of resources in a single administrative domain, taking into account the features of the jobs and resources, including the status of the network. However, only data associated with Grid jobs was considered-which is unlikely to be the case in practice.
We improve upon this by considering a more realistic way of checking the status of the network. We can say that this model will use feedback from resources and network elements in order to improve system performance. Scheduler will therefore adapt its behavior according to the status of the system, paying special attention to the system of network. The scenario is depicted in Figure 2 [1, 5] and has the following entities [1, 5].

- Users: Each one has a number of jobs to run.
- Computing Resources: May consist of single machine or clusters of machines.
- Routers:(write something)
- GNB (Grid Network Broker): An automatic network-aware scheduler.
- GIS (Grid Information Service): Which keeps a list of available resources?
- Resource Monitor: Which provides detailed information on the status of resources?
- BB (Bandwidth Broker): Which is in charge of administrative domain, and has direct access to the routers. BB can be used to support reservation of network links, and keep track of the interconnection topology between two end points within a network.

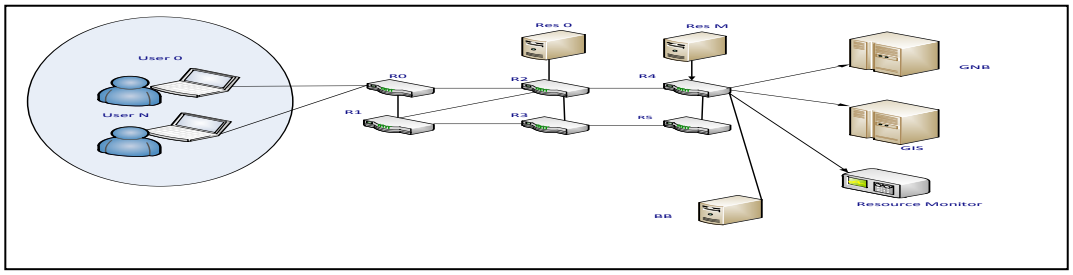

Figure 2: Grid scheduling Architecture

The interaction between components within the architecture is explained as represented in the Figure 2 as follows.

- Users ask the GNB for a resource to run their jobs. Users provide jobs and deadlines.
- The GNB performs two operations on each job. First it performs scheduling of that job to a computing resource, and second, performs connection admission control (CAC).
- The GNB makes use of the GIS in order to get the list of available resources, and then it gets their current load from the resource monitor.
- The GNB makes use of the BB in order to carry out operations requiring the network.
- Once the GNB has chosen a computing resource to run a job, it submits the job to that resource. On the completion of the job, the GNB will get the output sent back from the resource, and will forward to the user. Also, the GNB will update information about the accuracy of its decisions, considering CPU and transmission delays.

3. The Grid Scheduling

The term Scheduling can be understood as the allocation of machines or processors over time to perform a collection of tasks or as the problem of finding the optimal temporal assignment of some resources to certain tasks [6].

Scheduling is a way in which processes assigned to run on suitable systems in such a way the scheduling parameters are optimized. The scheduling parameters could be throughput, system utilization, and turnaround time, waiting time, response time, fairness or any other QoS parameter. Some of these parameters are discuss below [7].

- ***Throughput*** is measured as number of tasks executed per unit time. There are many possible throughput metrics depending on the definition of unit of work. Examples of throughput metrics at the resource layer include the effective transfer rate in Kbytes/sec under the Grid-FTP protocol and the number of queries/sec that can be processed by the database server of a law enforcement agency needed by the RAM application.
- ***System Utilization*** *is* to keep system as busy as possible.
- ***Turnaround time*** is estimated as the time taken by the job from its submission to the final execution. Thus, it is always expected from a computational grid scheduler to allocate the job to those grid resources which results in the faster overall execution of the job i.e. with minimum turnaround time.
- ***Waiting time*** *is* amount of time spend to wait by a particular job in system for getting a resource. In other words waiting time for a job is estimated as the time taken by the job from its submission to the get system for execution. The waiting time depend on the parameters similar as turnaround time.
- ***Response time*** is amount of time to get first response in time sharing system. The response time depend on the parameters similar as turnaround time.
- ***Fairness*** of system is defined as the time taken by each system in grid environment is same

3.1 Types of scheduling

Different types of scheduling [6, 7] are found in Grid systems as applications could have different scheduling needs such as batch or immediate mode, on the other hand, the Grid environment characteristics themselves impose restrictions such as dynamics, use of local schedulers, centralized or decentralized approach, etc. In order to achieve the desired performance, both the problem specifics and Grid environment information should be "embedded" in the scheduler. In the following, the main types of scheduling arising in Grid environments are described.

- **Static versus Dynamic Scheduling:** There are essentially two main aspects that determine the dynamics of the Grid scheduling, namely: (a) The dynamics of job execution, which refers to the situation when job execution could fail or, in the pre-emptive mode, job execution is stopped due to the arrival in the system of high priority jobs; and (b) The dynamics of resources, in which resources can join or leave the Grid in an unpredictable way, their workload can significantly vary over time, the local policies on usage of resources could change over time, etc. These two factors decide the behavior of the Grid scheduler, ranging from static to highly dynamic. For instance, in the static case, there is no job failure and resources are assumed available all the time. Although this is unrealistic for most Grids, it

could be useful to consider for batch mode scheduling: the number of jobs and resources is considered fixed during short intervals of time and the computing capacity is also considered unchangeable. Other variations are possible to consider the dynamics of resources and jobs.

- **Single Criterion versus Multi-criteria Scheduling:** If there is only one criterion to optimize, we are dealing with scheduling problems and models with a single criterion. Its characteristics are collected in the parameter if several criteria are considered, we are dealing with multi-criteria scheduling problems and models and the parameter *r* is replaced by a vector of parameters. Multi-criteria scheduling problems are usually more difficult than single criteria ones.
- **Centralized Scheduling:** Single job scheduler on one instance and all information are collected here. The scheduler is conceptually able to produce very efficient schedules, because the central instance has all necessary information on the available resources .Centralized scheduling is categorized into two forms.
- **Decentralized Scheduling**: No central instance is responsible for having information about state of all system and distributed schedulers interact with each other and decide the allocations for each job to be performed. Local job schedulers may have different but compatible scheduling policies .No communication bottleneck, scalable to greater extent, failure of single component doesn't affect whole meta-system and better fault tolerance and reliability than centralized systems Decentralized Scheduling also is of two types.

4. Issues and Challenges in Grid Scheduling

Now day's security is of critical importance for a wide range of real-time applications. Section 4 describes issues and challenges in scheduling the context of Grid environments in detailed [8].

4.1 Resource Management Issues

The resources that are coupled in grid computing environment are geographically distributed and different individuals or organizations own each one of them and they have their own access policy, cost, and mechanism. The resource owners manage and control resources using their favorite resource management and scheduling system (called local scheduler) and the grid users are expected to honor that and make sure they do not interfere with resource owners' policies. They may charge different prices for different grid users for their resource usage and it may vary from time to time. The global resource management and scheduling systems (e.g., Nimrod/G), popularly called grid schedulers or meta-schedulers, coordinate the user access to remote resources in cooperation with local schedulers(e.g., Condor and LFS) via grid middleware services [20,21]

4.2 Load Balancing in Grid Computing Environments

Load balancing is a computer networking method to distribute workload across multiple computational machine, network links, central processing units, disk drives, or other resources, to achieve optimal resource utilization, maximize throughput, minimize response time, and avoid overload. Using multiple components with load balancing, instead of a single component, may increase reliability through redundancy. The load balancing service is usually provided by dedicated software or hardware, such as a multilayer switch or a Domain Name System server. To minimize the time needed to perform all tasks, the workload has to be evenly distributed over all nodes which are based on their processing capacity. The Load Balancing problem is closely related to scheduling and resource allocation. It is concerned with all the techniques allowing and evenly distribution of the workload among the available resources in a system. The main objective of a load balancing consists primarily to optimize the average response time of applications: this often means the maintenance the workload proportionally equivalent on whole system resources. Load balancing is usually described in the literature as either load balancing or load sharing and load leveling [12].

The Load Balancing mechanism [12]can be broadly categorized as

- Centralized or Decentralized
- Dynamic or Static
- Periodic or Non-periodic

4.3 Fault Tolerant Issues

Since grids are highly dynamic in nature, so must handle failure in the resources and check how changes in the topology and computational capability of the grid resources affect the efficiency in terms of deadline of the tasks. In case of fault treatment, using replication and other techniques, it is interesting to develop a viable economic model that could provide an execution environment, which guarantees certain minimum cost and time even in the presence of failures, unlike present fault tolerant approaches where, usually, cost increases exponentially. The existing reliability and fault tolerance model may be augmented by considering scalability and security into consideration. Most of the approaches of fault tolerance are based on the prediction of failure probability of the resources in certain time interval. It is hard to achieve the resource failure prediction even with years of historic trace data. Hence, efficient techniques are required which do not base their decisions on the specific failure model and do not rely on the failure prediction accuracy [16].

4.4 Scalability support in hardware

Methods of adding more resources for a particular application fall into two broad categories, vertical scale up and horizontal scaling. In the past, the price difference between the two models has favored "scale out" computing for those applications that fit its paradigm, but recent advances in virtualization technology have blurred that advantage, since deploying a new virtual system over a hypervisor (where possible) is almost always less expensive than actually buying and installing a real one. Larger numbers of computers means increased management complexity, as well as a more complex programming model and issues such as throughput and latency between nodes; also, some applications do not lend themselves to a distributed computing model. Scalability support in hardware is bandwidth and latencies to memory plus interconnects between processing elements. The scalability can be measured by the capacity of the real-time distributed systems in the sense that how the quality of security and guarantee ratio can be scaled by adding additional nodes, memory, or processing power [8,9]

4.5 Security Issues

The security requirements of a task include how to specify the security of the task, the possible range of the security quality and the overhead of achieving some particular degree of security quality. The difficulty lies in measuring the quality of security of real-time tasks. To the best of our knowledge, no existing literature has directly addressed the issue of accurately estimating security overhead experienced by security-critical real-time applications, and this becomes a significant open issue in the development of real-time security-aware scheduling schemes. Moreover, another challenge is to design and implement real-time security-aware scheduling schemes, which can meet specific real-time and security requirements of applications executing in distributed systems. The ultimate goal of security-aware scheduling is to guarantee security constraints in addition to real-time requirements of tasks running in distributed systems [4, 5].

5. Security and Real-Time Requirements

Recently, real-time application with security requirements increasingly emerged in large scale distributed systems like Grids. The security-aware scheduling algorithms can improve security of real-time applications while maintaining a high level performance for Grids. An increasing number of real-time applications have security constraints because sensitive data and processing require special safeguards against unauthorized access. For example, a variety of real-time applications such as aircraft control systems running on parallel and distributed systems require security protections to completely fulfill their needs. As such, it is mandatory to deploy security services to guard security-critical applications running on Grids. Snooping, alteration, and spoofing are three common attacks in Grid environments and therefore, we consider three security services (authentication service, integrity service, and confidentiality service) to guard against common threats. Snooping, an unauthorized interception of information can be countered by confidentiality services. Alteration, an unauthorized change of information can be countered by integrity services. Spoofing, an impersonation of one entity by another can be countered by authentication services. With these three securities services in place, user can flexibility select the security services to form an integrated security protection against a diversity of threats and attacks in a Grid computing environment [10].

An Example of Security- Sensitive Real-Time Applications

In general the server performs the following security operations [10]on behalf of its clients.

- Establishes secure connections with business partners and back-end applications
- Applies and verifies digital signatures
- Authorizes access based on digital certificates
- Validates credentials in real time using public key infrastructure
- Encrypts and decrypts request and responses

5.1 A Security-Aware Middleware Model

Recently, real-time applications with security requirements increasingly emerged in large scale distributed system like Grids. However, complexities and specialties of diverse security mechanisms dissuade users from employing existing security services for their applications. A security middleware model (SMW) [10] by which security-sensitive real-time applications are enabled to exploit a variety of security services to enhance trustworthy executions of the applications. A quality of security control manager (QSCM), which is a centerpiece of the SMW model, is designed and implemented to achieve a flexible trade-off between overheads caused by security services and system performance, especially under situations where available resources are dynamically changing and insufficient. A security-aware scheduling mechanism, which plays an important role in QSCM, is capable of maximizing quality of security for real-time applications running on Grids.

5.1.1 Security Middleware model (SMW)

Middleware is software that sits between two or more types of software and translates information between them. It is used to solve computer clients' heterogeneity and distribution issues by offering distributed system services that have standard programming interface and protocols. The aim of security middleware (SMW) model is to meet security requirements of a variety of applications and improving performance of distributed real-time systems.

5.1.2 Architecture of the SMW Model

The SMW model consists of a user interface, a framework, low-level security service APIs, a quality of security control manager, and security middleware services (see in Figure 3). The SMW model provides two different types of user interfaces, namely, a professional user interface and a normal user interface. The professional user interface is an interface between developers (e.g. programmers) and applications being developed. An editor, a compiler and a debugger are essential components of the professional user interface. Programmers are allowed to directly access the low-level security service APIs, thereby efficiently constructing applications with various security functions. A normal user interface sits between a normal user and the framework. By using the normal user interface, usually and IDE (Integrated Development Environment), a normal user such as a system administrator can leverage the framework to readily create applications with security requirements [10].

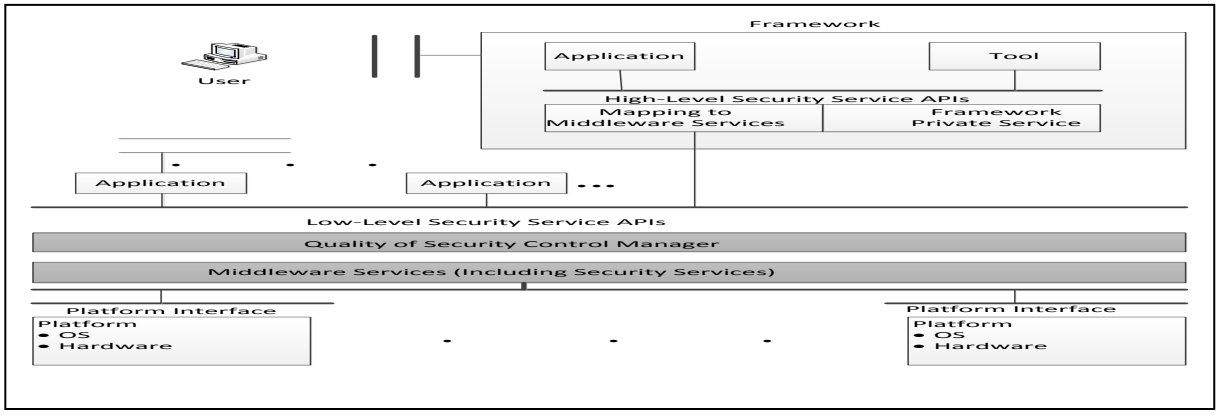

Figure 3: Security Middleware Architecture [10]

5.2 Security- Aware Scheduling Architecture

An m-node grid in which m identical nodes are connected via a high-speed network, e.g., Fast Ethernet, to process soft real-time tasks submitted by r users. Let N = {N_1, N_2....., N_m} denote a set of identical computational nodes. The architecture of security-aware real-time scheduling shown in Figure 4 encompasses the SAREC (Security-Aware Scheduling Strategy for Real-Time Applications on Clusters and Grids) strategy and a real-time scheduler. The SAREC strategy is implemented in form of a security level controller and an admission controller. A real-time scheduler using the EDF policy, which can be substituted by other real-time scheduling policies, is presented. The admission controller determines if an arriving task in a schedule queue can be accepted or not, whereas the security level controller aims at maximizing the security levels of admitted tasks [10].

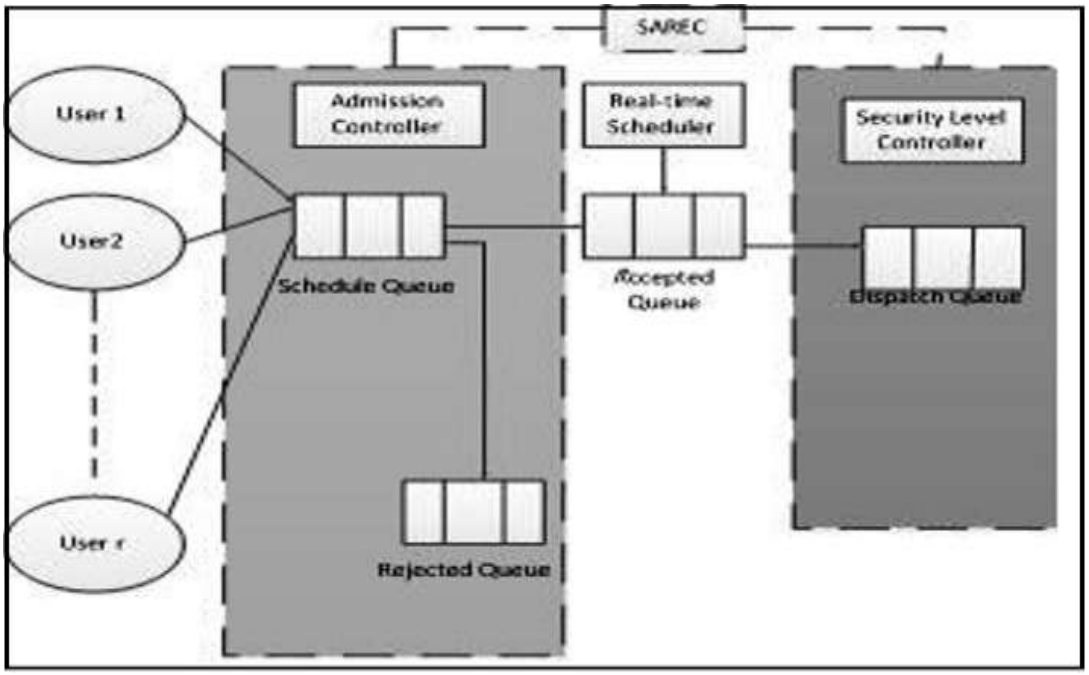

Figure 4: Security Aware Scheduling Architecture [11]

The schedule queue maintained by the admission controller is deployed to accommodate incoming real-time tasks. If the deadline and minimal security requirements of an incoming task can be guaranteed, the admission controller will place the task in an accepted queue for further processing. Otherwise, the task will be dropped into a rejected queue. The real-time scheduler processes all the accepted tasks by its scheduling policy before the tasks are transmitted into a dispatch queue, where the security level controller escalates the security level of first task under two conditions: (1) the security level promotion will not take the first task miss is deadline; and (2) increasing the security level will not make any previously accepted task miss its deadline. After being handled by the security level controller, the task is dispatched to one of the designated node Ni for all N referred to as a processing node for execution. For each processing nodes maintain a local queue [11].

6. Soft Computing Approaches

Soft computing approaches [20] are a large family of methods that have shown their efficiency for solving combinatorial optimization problems. Soft computing approaches usually require large running times if suboptimal or optimal solutions are to be found. However, the objective is to find feasible solutions of good quality in short execution times, as in the case of Grid scheduling, we can exploit the inherent mechanisms of these methods to increase the convergence of the method. Some popular soft computing approaches used in grid scheduling are as follows.

6.1 Genetic Algorithm

Genetic Algorithm is one approach from the various soft-computing approaches which can be used to find a solution in less time although it might not be the best solution. GA works on the basis of natural selection and evolution. The GA operators are applied over a randomly generated population which comprised of a set of chromosomes (solutions), in each generation. These chromosomes are evaluated against a fitness function derived on the basis of the optimization objective. The chromosomes that offer best finesses are selected for mating to reproduce offspring. This procedure is iterated over the generations resulting in good parents to reproduce better offspring. The process stops when the result converges. GA uses various operators to implement this operation which are as follows [19].

6.2 Ant Colony Optimization

Ant Colony Optimization **(**ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. Initially proposed by Marco Dorrigo in 1992 Ant Colony Optimization (ACO) studies artificial systems that take inspiration from the behavior of real ant colonies and which are used to solve discrete optimization problems. In the real world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. Each ant moves at random. Pheromone is deposited on path. Ants detect lead ant's path, inclined to follow. More pheromone on path increases probability of path being followed [20].

6.3 Particle Swarm Optimization

Particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Such methods are commonly known as meta-heuristics as they make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, meta-heuristics such as PSO do not guarantee an optimal solution is ever found [20].

7. Proposed work

Grid computing meant to enable the users in performing high performance computing in a collaborative and shared manner. The resources in grid are computational resources (nodes) as well as data, storage or something else for executing the job or applications available. Among all these resources, CPU is one with most importance when being concerned with the computational grid as these are responsible for effective execution of the job. Therefore, scheduling a job on them becomes a very challenging job with the problem being how to allocate the appropriate grid resources to the job demanding execution while primarily meeting the objective of high throughput and low latency. But the mapping of these resources on the jobs considering security requirements is not an easy job as the participating resources forming the grid are heterogeneous in nature with heterogeneity varying from type, size, management systems to management policies to name a few. In the work, we propose to develop security aware scheduling models to effectively schedule a job on the grid resources optimizing the quality of service parameters (e.g. Makespan, utilization, throughput, matching proximity, flow time, memory usage, security performance matrices, reliability, etc.) by using the scheduling heuristics (e.g. Min-Min, Max-Min, MET, MCT, etc.) , soft computing approaches as GA, ACO, PSO, etc and hybridized of them with considering the security aware scheduling architecture for the purpose of achieving the security requirements. The work will be finished by analyzing the performance of the models by comparing with similar models in the literature.

References

1. Ian Foster, Carl Kesselman, *The Grid 2: Blueprint for a Future Computing Infrastructure*. Second Edition ,Morgan Kauffman, 2006.
2. Uwe Schwiegelshohn , Rosa M. Badia , Marian Bubak ,(et. al.), Perspectives on Grid Computing . *Science Direct (ELSEVEIRE)*, 2010 .
3. Ian Foster, Carl Kesselman and Steven Tuecke, The anatomy of the Grid Enabling scalable virtual organizations, *International Journal of High Performance Computing Applications* 15 (3) (2001), pp. 200–222.
4. D. Kebbal, E.G. Talbi, J.M. Geib, "*Building and Scheduling Parallel Adaptive Applications in Heterogeneous Environments*", 1st IEEE Computer Society International Workshop on Cluster Computing, Melbourne, Australia, December 02 - 03, 1999.

5. Radulescu, Arjan J.C. van Gemund, "*Fast and Effective Task Scheduling in Heterogeneous Systems*", In Proc. of the 12th Euromicro conferences on Real-time Systems, pp229-238, 2000.
6. Baker, K.R., "*Introduction to Sequencing and Scheduling.*", John Wiley, 1974.
7. Xhafa, F., Abraham, A., "C*omputational Models and Heuristic Methods for Grid Scheduling Problems",* Future Generation Computer Systems, Elsevier, pp. 608-621, 2010.
8. Open Issues in Grid scheduling workshop. *National e Science centre*. http://www.nesc.ac.uk/technical_papers/UKeS-2004-03.pdf (Oct, 2003).
9. Abawajy, J. H., "*Fault-Tolerant Scheduling Policy for Grid Computing Systems*", Proceedings of the 18th international Parallel and Distributed Processing Symposium (IPDPS '04)
10. C.-J. Hou and K. G. Shin, "*Allocation of Periodic Task Modules with Precedence and Deadline Constraints in Distributed Real-Time Systems*", IEEE Trans. Computers, Vol. 46, No. 12, pp.1338-1356, Dec. 1997.
11. H. Topcuoglu, S. Hariri, M.-Y. Wu, "*Task Scheduling Algorithms for Heterogeneous Processors*", In Proc. of 8th Heterogeneous Computing Workshop, pp.3-14, 1999.
12. M.E. Thomadakis and J.-C. Liu, "*On the efficient scheduling of non-periodic tasks in hard real-time systems*" Proc. 20th IEEE Real-Time Systems Symp., pp.148-151, 1999.
13. Quan Liu, Yeqing Liao, "*Grouping-Based Fine-grained Job Scheduling in Grid Computing*", IEEE First International Workshop on Education Technology and Computer Science, vol.1, pp. 556-559, 2009.
14. T.F. Abdelzaher, E. M. Atkins, and K.G. Shin., "*QoS Negotiation in Real-Time Systems and Its Application to Automated Flight Control*" , IEEE Trans. Computers, Vol. 49, No. 11, Nov. 2000, pp.1170-1183
15. S. H. Son, R. Mukkamala, and R. David, "*Integrating security and real-time requirements using covert channel capacity*", IEEE Trans. Knowledge and Data Engineering, Vol. 12, No. 6, pp. 865 – 879, Nov.-Dec. 2000.
16. P. Koopman, "*Embedded System Security*" IEEE Computer, Vol. 37, No. 7, pp. 95-97, July 2004.
17. Irvine and T. Levin, "*Towards a taxonomy and costing method for security services*", Proc. 15th Annual Computer Security Applications Conference, pp.183–188, 1999.
18. M. Bishop, "Computer Security: Art and Science," *Addison Wesley Professional*, ISBN 0201440997; Published: Dec 2, 2002.
19. D. Goldenberg, **"***Genetic Algorithms in Search Optimization and Machine Learning*", Pearson Education, 2005.
20. http://en.wikipedia.org/wiki/Soft_computing