MODIFIED APPROACH FOR SECURING REAL TIME APPLICATION ON CLUSTERS

Abhishek Songra sw0512@mnnit.ac.in
Computer Science & Engineering Department
Motilal Nehru National Institute of technology, Allahabad, UP, India
Rama Shankar Yadav rsy@mnnit.ac.in
Computer Science & Engineering Department
Motilal Nehru National Institute of technology, Allahabad, UP, India
Sarsij Tripathi cs0620@mnnit.ac.in
Computer Science & Engineering Department
Motilal Nehru National Institute of technology, Allahabad, UP, India

Abstract

In today arena security critical real time applications running over clusters are growing very rapidly. As an application running on clusters demand both timeliness and security thus, an efficient scheduling algorithm is needed that have better performance in terms of both number of task accepted and security value received. This paper modifies the security aware scheduling approach [5] by utilizing the concept of task criticality and adaptive threshold value. Also, this paper discus the system architecture used, mathematical model, lemmas and modified scheduling approach. Further, simulation studies have been carried out in MATLAB (module for Real-time) to measure the performance of modified approach. The modified approach is applicable over wide range of application differing in there requirement and have better performance.

Keywords: Real time System, Scheduling, Security Services, Clusters

1. Introduction

A Real-time system is a system in which computations must satisfy stringent timing constraints besides providing logically correct results i.e. a correct computation of the result must finish before its specified deadline is met. Failure to meet the specified deadline in such system leads to catastrophic loss in case of hard real time systems whereas degraded performance is observed in soft real time application.

Many real time applications are using clusters for satisfying the need of high computing power where nodes are inter connected through high speed network. A real time applications using clusters faces security threats for example in stock quote update and trading system, incoming requests coming from different business partner while outgoing response from an enterprise back-end machine these application composed of clusters that has to satisfy both timeliness of response and security requirements [13]. As cluster executes vast number of unverified application submitted by vast number of different type of users both applications and users can be source of security threats to cluster [20]. These applications are vulnerable to attacks such as: attack by malicious user, malicious application running on clusters itself. The malicious users intercept applications running and launch denial of service whereas blocking of resources is observed in the case of malicious applications. The security threats to these applications are primarily related to the authentication, integrity, and confidentiality of application. An attacker may breach the above security service by spoofing, snooping and alteration kind of attack. These attacks are briefly defined below.

Spoofing attack is a situation in which one person or program successfully masquerades as another by falsifying data and thereby gaining an illegitimate advantage.

Snooping attack is not necessarily limited to gaining access to data during its transmission. Hacker may gain access to data while it is in transmission but can also gain access while the data is in not in transmission.

Alteration is a kind of attack in which a malicious user, which may be inside the cluster or outside the cluster, after gaining access to data performs unauthorized changes to it.

Application having real time constraints running over clusters requires secure computation. These applications have to satisfy both timeliness and security issues. Also, applications require preference of one security service to another one and different security services require overhead. Thus, an efficient scheduling algorithm is needed that achieves high performance in term of completing more number of computations while maintaining higher security level.

Rest of the paper is organized as follows. Section 2 deals with related work whereas system model along with modified scheduling approach are discussed in section 3. Section 4 includes simulation and result while paper is being concluded in section 5.

2. Related Work

Here, first we discuss related work done in the area of real time scheduling, followed by cluster based security issues and then proposed solution for problem. Extensive work has been done in the field of real time task scheduling whereas few work is reported on scheduling of real time tasks with security constraints. Based on the time of when scheduling decision, is taken scheduling algorithms are categorized as Offline (static) and Online (dynamic). In offline scheduling is performed well before system starts functioning however, scheduling decision are taken at run time in case of online. Authors in [8] have proposed an algorithm which schedules the task on uniprocessor systems whereas scheduling algorithm for multiprocessor system is given in [9] [11].

In [10] a non preemptive static scheduling algorithm is used whereas dynamic scheduling algorithm for multi-processor system is given in [11]. These algorithms did well for the real time systems but they fails to satisfy security constraints required for real time cluster based system.

T. Sterling and D. Savarese [14] used static scheduling on the clusters whereas dynamic scheduling approach is employed in [15].These works are focused for scheduling non real time tasks with security constraints on the multiprocessor systems and fail to satisfy the real time task requirement. Thus, Scheduling Real time task with security on clusters has become open area of research and few studies has been made in this area. Manhee Lee et. al. has discussed the security issues related with clusters [17] whereas grid computing discussed in [18].

Xie et. al. [5] has used a security aware scheduling strategy for real time applications on clusters to satisfy minimum security requirement. Scheduling decisions are taken based on earliest deadline first (EDF) [5]. Scheduling decisions are taken at two phase: first that satisfy the minimum security requirement while improvement in security is received in second phase. Authors [5] uses improvement in second phase on the basis of the arrival time, i.e., a job arrived later have lesser chance for improvement as compared to arrived earlier. The improvement on the basis of arrival time may lead to a situation that already feasible task in phase 1may rejected. This could be understood by an example given below.

Consider a task having attribute ($a_i, e_i, f_i, d_i, \mathcal{E}_i, S_i, \mathcal{L}_i$) where a_i, e_i, f_i, d_i, $\mathcal{E}_i$, S_i, $\mathcal{L}_i$ are the arrival time, execution time, finish time, deadline, amount of data to be secured, security level requirement and the criticality of a task respectively. Also, a task $\mathcal{T}_i$ requires q security services which are represented by set of security level ranges, e.g., $S_i = (S_i^1, S_i^2, \ldots\ldots, S_i^q)$ where S_i^j is security level range for j^{th} security service. The security criticality of a task is the cumulative security requirement of a task. A task is said to be more security critical if its security requirement is more than threshold value. Detailed security criticality will be explained in section 3.1.5. Consider set of two tasks ($\mathcal{T}_1, \mathcal{T}_2$) having attributes value as below.

Tasks requiring three security services (authentication, confidentiality and integrity) along with security level range, given in square brackets. For task $\mathcal{T}_1$ minimum authentication security level required is 0.2 and this is compared with security level and corresponding overhead given in Table 5. Incase security requirement does not directly match with table value next higher security level is being considered. For this authentication requirement (0.2) is not matched with the value given in the table so, next higher value (0.55) is considered and corresponding overhead is computed as authentication overhead as 90. Similarly minimum confidentiality (0.3) is selected from Table 3 as 0.36 with overhead is 5.33ms (200/37.5). Table 4 is used to determine integrity overhead. Similarly we can determine overheads of the three services for task $\mathcal{T}_1$. Finish time of a task is the sum of security overhead, execution time of $\mathcal{T}_1$ and waiting time due to higher priority task. The values are summarized in table 1(a) below.

Table 1(a): Feasibility of task set after phase one

Task	Authentication overhead (Min)	Confidentiality overhead (Min)	Integrity overhead (Min)	Finish time (e_i+Overhead+ w_i)	Deadline

$\mathcal{T}_1$	90	5.33	8.368	107.701	150
$\mathcal{T}_2$	90	4	12.5	216.201	222

It is clear from the table that both task are feasible with minimum securities after phase 1.In second phase author[5] consider task $\mathcal{T}_1$ for improvement as its arrival time is earlier than T2 .Finish time of T1 after improvement in services (authentication, confidentiality and integrity are 0.5,0.5 and 0.4 respectively) is 124.033 ms However ,finish time of $\mathcal{T}_2$ become 232.533 ms which is more than its deadline leading to rejection of $\mathcal{T}_2$. That is either both tasks are forced to run with minimum security or $\mathcal{T}_2$ will be rejected shown in table 1(b).

Table 1(b): Feasibility of task set after phase two with existing approach

Task	Authentication overhead (security value)	Confidentiality overhead (security value)	Integrity overhead (security value)	Finish time (e_i+Overhead+ w_i)	Deadline
$\mathcal{T}_1$	90 (0.5)	9.483 (0.5)	20.55 (0.4)	124.033	150
$\mathcal{T}_2$	90(0.3)	4(0.2)	12.5(0.3)	232.533	222

In this paper we modify criteria for selecting candidate task for the security improvement phase by using the concept of task criticality other than its arrival time. For purpose of adaptation between improvement in security and reduction in rejection of task, a threshold is considered. The value of threshold is determined dynamically, i.e., incase rejection is more the higher threshold value is taken; improvement in security is less consequently rejection ratio may be reduced. The next section deals with system model followed by modified approach.

3. System Model

This paper uses on line scheduling approach which is targeted for real time applications having security requirements on clusters. Cluster is a group of N nodes {N_1, $N2$, $N3$ ….N_n) connected through a high speed network where real time application having high computational and security requirements are submitted. These applications due to their high computational demands are incapable of executing on a single node; hence they are partitioned into sub application or tasks. For simplicity we presume that the tasks incorporated in an application are independent of each other. Real time application is accepted if and only if the cluster can schedule the task so that they complete within their respective deadline and ensures for at least minimum security requirement (related to application) in phase 1. Improvement over minimum security guarantee may be achieved through utilization of available slack in schedule. We consider a task set having n tasks, T = { $\mathcal{T}_1, \mathcal{T}_2$ …… $\mathcal{T}_n$}. Each task $\mathcal{T}_i$ is described with the attribute ($a_i, e_i, f_i, d_i, \mathcal{E}_i, S_i, \mathcal{L}_i$) where a_i is the arrival time, e_i is the execution time, f_i is the finish time d_i is the deadline, $\mathcal{E}_i$ is the amount of data to be secured, S_i is security level requirement, $\mathcal{L}_i$is the criticality of a task. Suppose a task $\mathcal{T}_i$ requires q security services which are represented by the security level ranges e.g. $S_i = (S_i^1, S_i^2, \ldots\ldots, S_i^q)$. The parameter and assumptions are same as used in [5].

Before we proceed for modified scheduling algorithm in detail, we first discuss the various terms used in this paper. These terms are summarized in Table 2.

Table 2: Terms and Description

Term	Description
m	Number of nodes in the cluster. The nodes may be or may not be identical.
R	Number of users submitting tasks to the cluster. A user can submit any task at any point of time.
e_i	Execution time of a task $\mathcal{T}_i$.
a_i	Arrival time of the task $\mathcal{T}_i$.
d_i	Deadline of task $\mathcal{T}_i$. It is the time beyond which the utility of the result of the task degrades.
f_i	f_i is the allowable finish time of the task $\mathcal{T}_i$ by which the utility of the result is within acceptable quality of service.
$\mathcal{L}_i$	Criticality of the task $\mathcal{T}_i$.

sl_i	Security level value assigned to a security algorithm based on its performance.
S_i	Security level of task T_i
$£_i$	Amount of data that is to be secured.
Th	Criticality Threshold of the cluster.
rej_ratio	Rejection ratio.
Min(rej_ratio)	Minimum rejection ratio. It is a measure of quality of service of the cluster that must be maintained.
Max(rej_ratio)	Gives the extreme limit of tasks rejection in percentage.

As snooping, alteration and spoofing are three common attacks on cluster that can be handled by security services such as Authentication, Integrity and Confidentiality. These services incurred computational overhead, which depends upon amount of data secured used for securing these attacks. The following sub section describes detail about these services along with mathematical model for computation of overhead as used in [5].

3.1 Security Overhead Model

This paper focused on deploying security services (authentication, integrity and confidentiality) to secure cluster based real- time application against the basic attacks (spoofing, snooping and alteration). Snooping, an unauthorized interception of information can be tackled by confidentiality service whereas authentication service is deployed for spoofing. The alteration is unauthorized modification to information; this can be taken care by integrity services. Different applications require different type of integration of these security services for example; one may weight these services of equal importance whereas other may weight one service over another one. Thus, different combination of these services leads to complex integration of these services. The security aware scheduler running over complex integration has to adapt security overhead experience by a task in order to achieve desired quality of services (QoS) may be measured as number of tasks accepted, cumulative security level etc. Similar type of consideration is used in [5]. The security services are independent of one another. . The user can select different security services from the available services to form a complex integrated security solution. The following paragraph discusses detailed mathemetical model for confidentiality followed by integrity and then authentication.

3.1.1 Confidentiality Overhead:

Confidentiality is achieved by encrypting & decrypting both real time application as well as data to receive safegaurd from malicious user. We consider eight standard encryption algorithms to calculate confidentiality overhead which is shown in Table 3 where each security algorithm is assigned a security level in the range of 0.08 to 1 on the basis of its security performance. Beside these security algorithms (given in table) security of other algorithm security overhead is calculated with the use of equation 1.

$$sl_i^c = \frac{13.5}{v_i^c}, 1 \leq i \leq 8 \qquad (1)$$

where v_i^c is performance of the i^{th} ($1 \leq i \leq 8$) standard encryption algorithm and sl_i^c is the confidentiality security level of task T_i.

The security level of a algorithm is inversely proportional to algorithm's performance.

$$sl_i^c \propto 1/v_i^c$$

In case required confidential security level of of task T_i is S_i^c, the overhead for this service can be computed by the use of equation 2 where $£_i$ is the amount of data (in terms of Bytes/KB/MB) which is to be secured & $\sigma^c(S_i^c)$ is a function used for mapping a security level to its corresponding encryption algorithm's performance.

$$c_i^c(S_i^c) = \frac{£_i}{\sigma^c(s_i^c)}, 1 \leq i \leq 8 \quad (2)$$

Table 3: Cryptographic Algorithms for Confidentiality Service

Cryptographic Algorithms	*SL_i^c: SL Security level*	*ν_i^c:KB/ms*
Seal	0.08	168.75
RC4	0.14	96.43
Blowfish	0.36	37.5
Knufu/Khafre	0.40	33.75
RC5	0.46	29.35
Rijndael	0.64	21.09
DES	0.90	15
IDEA	1.00	13.5

3.1.2 Integrity Overhead:

Integrity security service is used to guard data against unauthorized modification or tampering while task is executing. We consider that seven integrity algorithms are deployed for providing integrity service and these consideration are same as considered in [5]. Integrity is achieved by implementing hash function [24] where each function is assigned a security level in accordance with its performance. The hash functions are shown in Table 4 along with their respective performance & security level.The security level for other hash function except shown in table, can be computed from equation 3.

$$sl_i^g = \frac{4.36}{\nu_i^g}, 1 \leq i \leq 7 \quad (3)$$

Where ν_i^g is the performance of the i^{th} ($1 \leq i \leq 7$) hash function.

Table 4: Hash Function for Integrity Service

Hash Function	SL_i^g: Security level	ν_i^c:KB/ms
MD4	0.18	23.90
MD5	0.26	17.09
RIPEMD	0.36	12.00
RIPEMD-128	0.45	9.73
SHA-1	0.63	6.88
RIPEMD-160	0.77	5.69
Tiger	1.00	4.36

Let S_i^g is the security level of integrity service for task $\mathcal{T}_i$, the overhead due to integrity service can be computed using equation 4.

$$c_i^g(S_i^g) = \frac{£_i}{\sigma^g(s_i^g)}, 1 \leq i \leq 7 \quad (4)$$

where $£_i$ is the amount of data whose integrity is to be assured and $\sigma^g(S_i^g)$ is a function used for mapping a security level to its corresponding hash function's performance.

3.1.3 Authentication Overhead:

Authentication is used to tackel spoofing attack. The authentication service insured that all task must be submitted by authorized users. Three authentication methods are used in paper which is shown in Table 5 where each authentication method is assigned a security level value. Security level of a required authentication method (other than given in table 4) can be calculated using equation 5.

$$sl_i^a = \frac{\nu_i^a}{163}, 1 \leq i \leq 3 \quad (5)$$

where v_i^a is the performance of ith $(1 \leq i \leq 3)$ authentication method .
Authentication overhead $c_i^a(S_i^a)$ of task T_i is a function of T_is security level S_i^a.

Table 5: Authentication Methods for authentication service

Authentication Methods	SL_i^a: Security Level	v_i^c Computation Time(ms)
HMAC-MD5	0.55	90
HMAC-SHA-1	0.91	148
CBC-MAC-AES	1	163

3.1.4 Security Overhead Model:

The overall security overhead for task T_i which is the sum of overhead incurred by each of the three security services imployed in forming the integrated security solution , can be computed using equation 6. Consider a task T_i requires w security services in sequential order and s_i^k and c_i^k be the security level & security overhead of the kth security service applied on the task respectively. The overall security overhead of task can be calculated using equation 6.

$$c_i = \sum_{j=1}^{t} c_i^j(s_i^j), where\ s_i^j \in S_i^j \quad (6)$$

3.1.5 Security Criticality

The term security criticality is extracted from security services ranges for a given task and it is cumulative security requirement of task for different security services. For example already considered in section 2 the security criticality of task T_i is the average of lowest limit of the range for three security services ,i.e., security criticality of task T_1 ($\mathcal{L}_1$) is (0.2+0.3+0.1)/3 = 0.2 and $\mathcal{L}_2$ for T_2 is 0.2667 . Thus T_2 is more security critical than T_1.

3.2 System Architecture Used

System architecture used in this paper consist of 'm' identical nodes connected through a high speed network, where real time task submitted by the 'r' number of users is shown in Figure 1. The schedule queue maintained by admission controller is a buffer used to hold newly arrived task without any consideration. The task submitted by the user is dispatched to the accepted queue if it pass acceptance test. A task is said to be pass acceptance test if task is able to complete in its deadline with minimum security requirement. This acceptance test is the responsibility of admission controller. A task fail to pass the acceptance test is said to be rejected and such task are places to the rejected queue. In contrast to acceptance test performed by admission controller (where acceptance test of individual task is taken into account) real time scheduler performed feasibility analysis of newly accepted task along with other task waiting for service or partially executed. A task passes feasibility analysis join dispatch queue where security enhancement is achieved (phase 2). A task fail to satisfy feasibility test join rejected queue and accepted task is dispatched to local queue of nodes in cluster. Similar type of system architecture is used in [5].

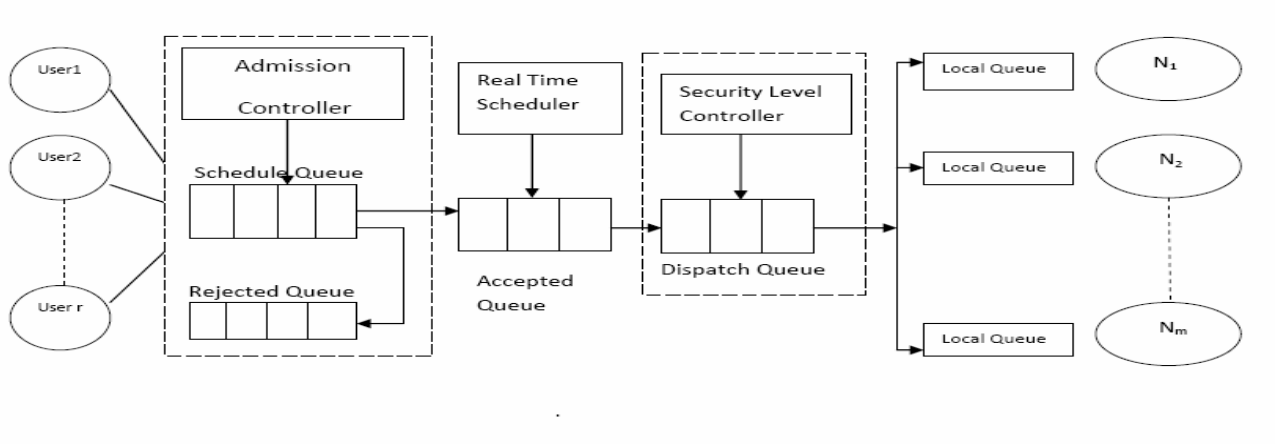


Fig.1 System Architecture Used

An application submitted to the cluster has the following property.

Prpoerty1 This paper considers hard real time application submitted to the cluster .The application is composed of 'n' independent tasks requesting different level of security. An application is said to be accepted if and only if all tasks are feasible. Each node estimates the wait time w_i^j of $\mathcal{T}_i$ on node N_j, will be the sum of remaining time of the executing task interrupted and execution time of all the tasks of higher priority, thus, $w_i^j = \text{remaining\_time}_h + \sum_{h \in H} e_h + \text{cost}(\min\_S_i)_i^j$ where H refers to the set of higher priority tasks (having deadline earlier to that of task $\mathcal{T}_i$).

After estimation of waiting time on a node, cost of the minimum security level feasibility analysis have been performed to obtain a valid schedule. A valid schedule can be stated by the following lemma used in [5].

Lemma 1 A valid schedule is the one in which the incoming task can be scheduled on at least one node on the cluster such that it can be granted minimum security guarantee without missing its own deadline nor forcing any previously accepted task to miss its respective deadline. Mathematically it is written as,

$$\exists N_j \in N \text{ such that } w_i^j + e_i + \text{cost}(\min\_S_i)_i^j \leq d_i \quad \text{(i)}$$

$$\forall \mathcal{T}_k \in \text{local queue to } N_j \text{ and having lower priority than arriving task } \mathcal{T}_i: w_k^j + e_i + \text{cost}(\min\_S_i)_i^j \leq f_k \quad \text{(ii)}$$

Where e_i, w_i^j are the worst case execution time, estimated wait time of the task $\mathcal{T}_i$ on node N_j respectively. The f_k is the allowable finish time of the task $\mathcal{T}_k$ by such that utility of the result is within acceptable quality of service.

Proof: If a task misses its own deadline then the utility of the result is lost. If it forces any previously accepted task to miss its deadline then an entire application will fail as refer property 1. In case, a task is accepted its security guarantee is improved in the best effort way if and only if the criticality of the task is more than the threshold of the cluster. This threshold is dynamically adjusted to maintain a desired QoS (rejection ratio not more than the value allowed for it) on the cluster i.e. to provide lower rejection ratio by allowing more tasks to be accepted by increasing the threshold. This can be stated as the following lemma.

Property 2: The estimated waiting time of a task $\mathcal{T}_i$ is given as $w_i^j = w_i^j + e_h + \text{cost}(S_h)_h^j$ where e_h and $\text{cost}(S_h)_h^j$ are execution time and overhead of security on node j respectively of task $\mathcal{T}_h$ (such that $d_h < d_i$) and its arrival time is a_h ,i.e., the task $\mathcal{T}_i$ may have to wait more than its estimated time because of the arrival, of a higher priority task before it can be scheduled. The estimated wait time of w_i^j can be given as $\sum_{h \in H} \text{actual\_finish\_time}_h^j \leq w_i^j \leq \sum_{h \in H} \text{estimated\_finish\_time}_h^j$ where H refers to set of higher priority tasks (having deadline earlier than the task $\mathcal{T}_i$), $\text{actual\_finish\_time}_h^j$ refers to the exact time by which the higher priority

task actually completes (by taking execution time between best and worst case), the $estimated\_finish\_time_{ij}$ refers to the execution time expected to be taken by the task in worst case.

Lemma 2: Threshold of a cluster is proportional to the rejection ratio on the cluster.

Proof: The value of the threshold of the cluster can lie between 0 and 1. If the value of threshold is equal to zero it indicates that all tasks will be improved in the best effort way at the time of acceptance, hence each task will demand maximum security overheads and will take at higher computation time. In such case, less number of tasks can be accepted. If the value of threshold is one then all tasks will be accepted with minimum security overhead and will be improved later at the time of their execution (if slack for improvement exists), hence more number of tasks can be accepted hence, lowering the rejection ratio. Thus threshold is directly proportional to rejection ratio.

3.3 Modified Security Aware Scheduling Approach (MSASA)

In [5] authors have used improvement in the security of a task on the basis of first come first service and reject a tasks whose minimum security requirement is not satisfied. As a result the scheme faces higher rejection ratio and lesser improvement in security too. In this paper beside given preference on the first come first service basis we schedule task with earliest deadline first to satisfy minimum security requirement. However, in improvement phase preference is given to more critical task (measured in terms of security requirement)
The security benefit received by a task is measured using security level function is given by equation 7.

$$SL(s_i) = \sum_{j=1}^{q} w_i^j s_i^j, 0 \leq w_i^j \leq 1, \sum_{j=1}^{q} w_i^j = 1 \quad (7)$$

where X_i denotes all possible schedules for task $\mathcal{T}_i$ and $x_i \in X_i$ is a scheduling decision for $\mathcal{T}_i$. For a given a real time task $\mathcal{T}_i$, the security benefit is maximized by security level controller using the following security benefits (SB), security value (SV) constraints as given below:

$$SB(X_i) = \max_{x_i \in X_i} \left\{ \sum_{j=1}^{q} w_i^j s_i^j(x_i) \right\} \quad (8)$$

The security level of task is increased up to a level at which task completes with in its deadline and does not make any previously accepted tasks to miss their deadline. The following security value function needs to be maximized under certain timing and security constraints:

$$SV(X) = \max_{x_i \in X_i} \{\sum_{i=1}^{p} y_i SB(x_i)\} \quad (9)$$

Where, p is the number of submitted tasks, y_i is set 1 if the task is accepted and is set to 0 otherwise. Our aim is to schedule tasks, while maintaining the guarantee ratio, in a way to maximize equation 10.

$$SV(X) = \max_{x \in X} \{\sum_{i=1}^{p} (y_i \max_{x_i \in X_i} \{\sum_{j=1}^{q} w_i^j s_i^j\})\} \quad (10)$$

After the possible improvement in the task's security level it is dispatched to node accepting it and promising the best security level or minimum wait time (if less critical).The modified security aware scheduling algorithm is given below.

Improve_security ()

Arrange security services according to their weights
  For each security services do
    Calculate overhead for S_i^k for kth security service C_i^k
    EFTij= wij + e_i + C_i^k
    If (EFTij > Di)
      Decrease S_i^k break

Increase S_i^*

Continue till security level of all security services is not maximized

MSASA Algorithm ()

//Input: Task to be scheduled with their security requirements

//Output: Tasks are scheduled on nodes.

For every task T_i arriving into schedule queue.

For every node N_j do

Calculate wait time of T_i on N_j is w_i^j

Calculate cost of T_i on N_j is c_i^j (min (SL))

Estimated finish time $EFT_i^j = w_i^j + e_i + c_i^j$ (min (SL))

If ($EFT_i^j < d_i$)

$Accept_i^j$=1 on the node N_j

Else

$Accept_i^j$=0 on node N_j

If ($accept_i^j$ ==1 && criticality of T_i > threshold Th)

Call improve_security ()

If task is accepted on any node then

Increase accepted task

Select the best node for scheduling T_i (let it be N_k)

If (my_id==k)

Insert the task T_i in local queue based on EDF

Delete the task T_i from the arrive queue

Else

Increase rejected_task

Rejection_ratio = rejected_task / (rejected_task+accepted_task)

if (rejection_ratio > MAX(rejection_ratio))

Increase threshold Th

Else

Decrease threshold Th

Continue with next task if any

Let us consider task T_1 and T_2 used in the section 2.Now we will examine the effect of the modified approach on these two tasks. As we know task T_2 is more security critical than T_1 and in section 2 from table 1(a) it is clear that both task are schedulable with minimum security requirements whereas from table 1(b) it is evident that improving security of task T_1 causes task T_2 to miss its deadline. By our modified approach the task T_1 is accepted at minimum security requirement and security improvement is done in task T_2. These results are shown in the table 5.

Table 5: Feasibility analysis after improvement phase with modified approach

Task	Authentication overhead(security value)	Confidentiality overhead(security value)	Integrity overhead(security value)	Finish time (e_i+Overhead+ w_i)	Deadline
T_1	90(0.2)	5.33(0.3)	8.368(0.1)	107.701	150
T_2	90(0.55)	5.11(0.46)	15.416(0.45)	220.227	222

4. Performance measurement and discussion: The performance of modified security aware scheduling approach (MSASA) is measured through simulation in MATLAB environment using scheduling tool. The simulation

parameters used in this paper is same as used in [5] and are summarized in table 6. The performance of MSASA is compared with that of security aware scheduling approach (SASA) [5]. The key parameters are guarantee ratio (ratio of number of tasks accepted over total number of tasks arrived in the system) and security value received (sum of achieved security for the entire accepted task).

Table 6: Simulation Parameters

Parameter	Value (Fixed)-(Varied)
β (Deadline base, or Tbase)	(0ms) - (10,50,100……800)ms
Execution time e_i	Uniform random number [5, 20].
Required Security Service	(Mixed)- (confidentiality only, Integrity Only, Authentication Only)
Weight of Authentication	(0.2)- (0.1,0.3)
Weight of Confidentiality	(0.5)- (0.1,0.2……0.8)
Weight of Integrity	(0.3)- (0.1,0.2……0.8)
Threshold	(0.5)- (0.1,0.2…….1)

Generation of task set:
Task has Poisson distribution arrival pattern with execution time uniformly generated. The range of security services are chosen by selective uniform random number between 0.1 to 1.0.

We used the following equation to generate $\mathcal{T}_i$ *s* deadline d_i.

$$d_i = a_i + e_i + cost_i^{max} + \beta \quad \textbf{(11)}$$

Where, a_i= arrival time of task, e_i= execution time of task and $cost_i^{max}$ is maximal security overhead which is computed as follows:

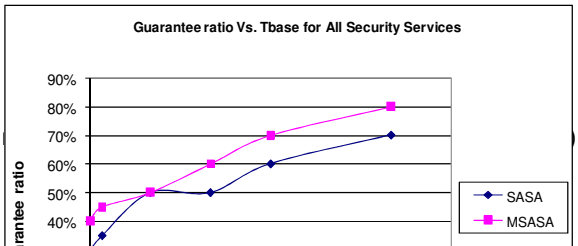
$$cost_i^{max} = \sum_{j \in \{a,c,g\}} cost_i^j \left(\max\{S_i^j\}\right) \quad \textbf{(12)}$$

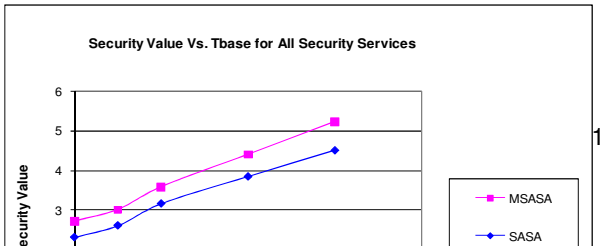
Where, $cost_i^j \left(\max\{S_i^j\}\right)$ represents the overhead of the j^{th} security service for $\mathcal{T}_i$ when the corresponding maximal requirement is satisfied.

4.1 Results and discussion:

Simulations results are obtained in variety of applications requesting different type of services with different security levels. In following section we first discuss effect of Tbase for application where all there security requirements are needed followed by application requesting only special kind of security.

Effect of Tbase for all security requirements: Figure 2 (a) and 2 (b) shows performance of modified security aware scheduling approach for the case where authentication, integrity and confidentiality services are required. Figure 2 (a) measured the performance in term of guarantee ratio whereas security value is measured in Figure 2(b) .It is observed that with increment in Tbase both guarantee ratio and security value increases but this increment in performance is more in MSASA as compared to SASA. This is because increment in Tbase deadline of a task relaxed giving better performance in both cases. However , in improvement in rejection ratio by the use of threshold we decrease the number of task whose security is improved and accept more number of tasks with minimum security this gives better performance in both terms as compared to that received incase of existing one.



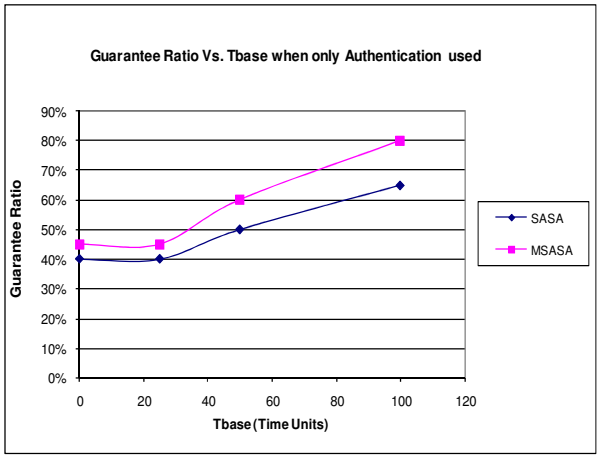


.

Fig 2(a): Effect of Tbase.

Fig 2(b): Effect of Tbase

Effect of Tbase for authentication service only: Performance of MSASA is shown in Figure 3(a) and 3(b) for the case where application request for authentication services. It is observed that guarantee ratio of modified approach increases with increment in Tbase value. However, security value received is almost same both of the approach.



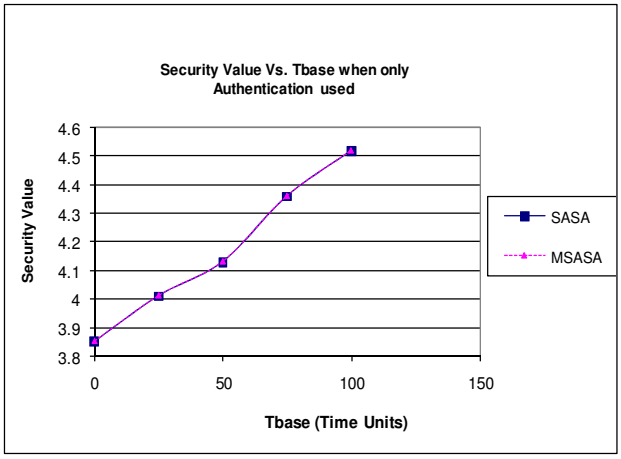
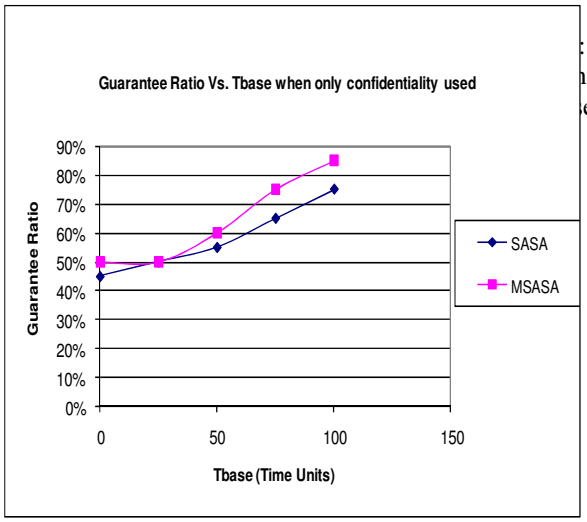


Fig3 (a): Impact of the Authentication service

Fig 3(b): Impact of the Authentication Service

: Figu
hat mod
erved r



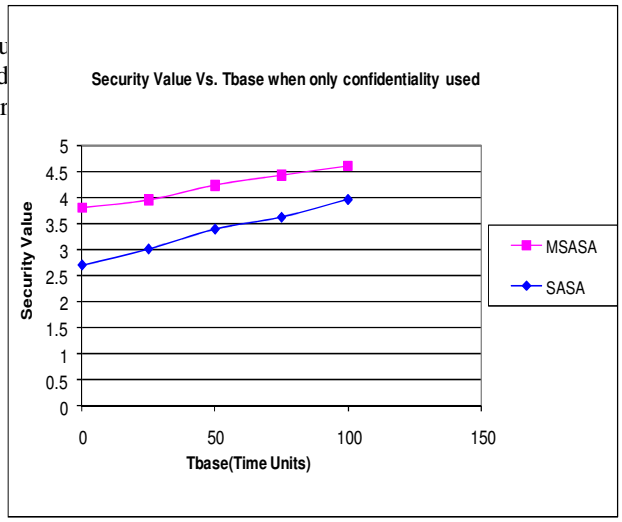


Fig 4(a): Impact of the Confidentiality Service

Fig4 (b): Impact of the confidentiality Service.

Effect of Tbase for integrity services only: The impact of integrity service is shown in figure 5(a) and 5(b). Similar type of trained is obtained as observed in the case confidentiality service only.

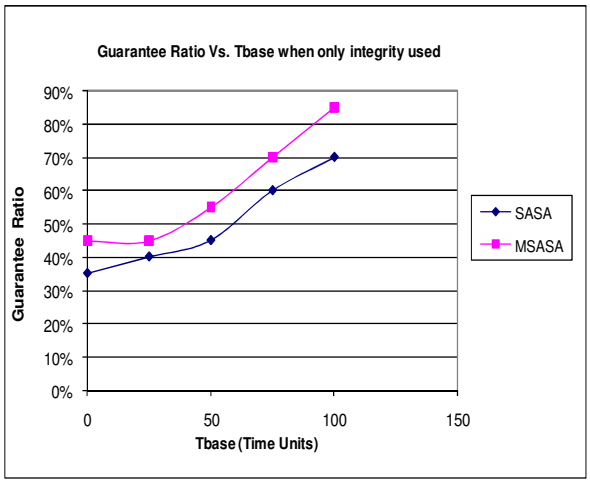


Fig 5(a): Impact of the integrity Security Service

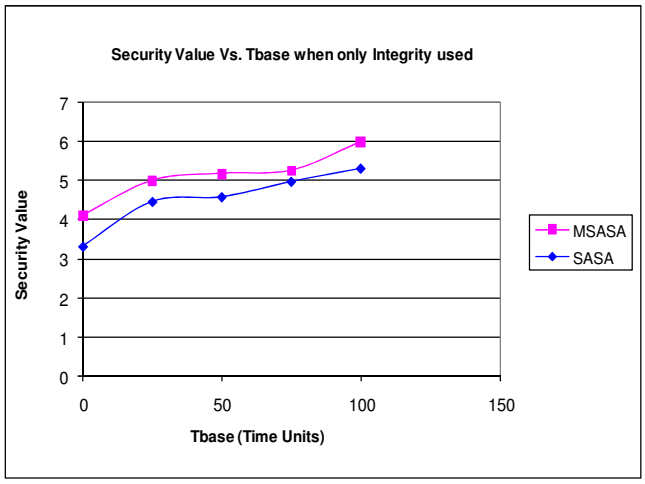


Fig 5(b): Impact of the integrity Security Service

5. Conclusion:

Security and timeliness both are equally important parameter for real time applications running over clusters. In this paper we propose a modified security aware scheduling approach that utilizes the concept of criticality and threshold based improvement in security of task over its minimum security requirement. This paper discusses system architecture, mathematical modeling and modified approach. The performance of modified approach is observed to simulation studies and example used. It is observed that modified approach have improvement about 15 % in terms of both guarantee ratio and security value received. The modified approach is applicable over wide range of application requesting different kind of security services and trimming constraints.

References:

1. Makan Pourzandi, Ibrahim Haddad, Charles Levert, MiroslawZakrzewski: A New Architecture for Secure Carrier-Class Clusters. IEEE International Conference on Cluster Computing, 23-26 Sept. 2002, Page(s):494 – 497.
2. Dessouly, Alaa Amin and Reda Ammar and Ayman El: Scheduling Real Time Parallel Structures on Cluster Computing with Possible Processor Failure. IEEE 9th International Symposium on Computers and Communications, Volume 1, 28 June-1 July 2004, Page(s):62 – 67.
3. Parnas, J.Xu and D.L.: Scheduling Processes with Release Times, Deadlines, Precedence and Exclusion Relations. Transactions on Software Engineering, IEEE Volume 16, Issue 3, March 1990, Page(s):360 - 369
4. SHILOH, O. t. Amnon BARAK: Scalable cluster computing with MOSIX for LINUX. In Proceedings of 5th Annual Linux Expo, pages 95--100, May 1999.

5. Qin, Tao Xie and Xiao: Scheduling Security Critical Real Time Applications on Clusters. IEEE transactions on computers, Vol. 55, no 7, pp. 864-879 July 2006.
6. Gagne, T.Shepard and M: A Pre-Run-Time Scheduling Algorithm for Hard Real Time Systems. Transactions on Software Engineering, IEEE Volume 17, Issue 7, July 1991, Page(s):669 - 677.
7. X. Zhang, Y. Qu, and L. Xiao.: Improving Distributed Workload Performance by Sharing both CPU and Memory Resources. 20th International Conference on Distributed Computing Systems, IEEE. 10-13 April 2000, Page(s):233 – 241.
8. Kavi, Wenming Li and Krishna: A Non Preemptive Scheduling Algorithm for Soft Real Time Systems. Computers & Electrical Engineering, Volume 33, Issue 1, January 2007, Pages 12-29.
9. O.Elkeelany, M.matalgah, K.Sheikh: Performance Analysis of IPSEC Protocol: Encryption & Authentication. International conference on Communication IEEE 2002. Volume 2, page(s):1164-1168.
10. Martel, K. Jeffay and C. U: On Non-Preemptive Scheduling of periodic and Sporadic Tasks: Proceedings of the 12th IEEE Real-Time Systems Symposium, San Antonio, Texas, December 1991, IEEE Computer Society Press, pp. 129-139.
11. M. L. Dertouzos and A. K Mok : Multi-Processor Online Scheduling of Hard Real- Time Tasks: IEEE Transactions on Software Engineering, Vol. 15, No. 12, December 1989 , pp. 1497-1506.
12. J.Deepkumara, H.M. Heys and R.venkatesan: Performance Comparison of Message Authentication Code for Internet protocol Security. www.engr.mun.ca/~howard/PAPERS/necec_2003b.pdf 2003.
13. Genesis, M. H. A.M. Goscinski and J. Silock: The operating system managing parallelism and providing single system image on cluster. LNCS volume 2790/2004, publisher Springer Berlin / Heidelberg.
14. Savarese, T. Sterling and D : A parallel workstation for scientific computation. Proceedings of the 24th International Conference on Parallel Processing, August 14-18, 1995, Urbana-Champain, Illinois, USA. Volume I: Architecture.
15. A J.Hong, X. Tan and D. Towsley: performance analysis of minimum laxity and earliest deadline scheduling in a real time system. IEEE Transactions on Computers, Volume 38, Issue 12, Dec. 1989, Page(s):1736 - 1744.
16. Foster, Ian, Nicholas Karonis: Managing Security in High Performance Distributed Computations. Journal of Cluster Computing Volume 1, Issue 1 pages 95-107, publisher Springer Netherlands 1998.
17. Manhee Lee, Eun Jung Kim, Ki Hwan Yum: An overview of security issues in cluster interconnects. Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops, 2006. Volume 2, 16-19 May, Page(s):9 pp.
18. Ian Foster, Carl Kesselman, Gene Tsudik, Steven Tuecke : A security architecture for computational grids: Proceedings of the 5th ACM conference on Computer and communications security CCS 1998.
19. Ferrari, Adam et al. A flexible security system for Metacomputing Environments www.cs.virginia.edu/papers/hpcn99.pdf 1999.
20. R. David, S. Son and R. Mukkamala: Supporting Security Requirements in Multilevel Real Time Database. IEEE Symposium on Security and Privacy, 8-10 May1995, Page(s):199 – 210,.
21. R. Mukkamala and S. Son: A Secure Concurrency Control Protocol for Real-Time Database: IFIP Workshop on Database Security. 1995.
22. S.H. Son, C. Chaney, C. and N. Thomlinson: Partial Security policy to Support Timeliness in Secure Real Time Databases. IEEE Symposium on Security and Privacy, 3-6 May 1998, Page(s):136 – 147.
23. Bosselares, R.Govaerts : Fast Hashing on the Pentium. Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology. Springer-Verlag, 1996 Pages: 298 - 312.