This paper appeared in the *Proceedings of the 7th IEEE International Conference on Cluster Computing* (Cluster 2005), Boston, USA, September 27-30, 2005.

A New Allocation Scheme for Parallel Applications with Deadline and Security Constraints on Clusters

Tao Xie Xiao Qin
Department of Computer Science
New Mexico Institute of Mining and Technology
Socorro, New Mexico 87801
{xietao, xqin}@cs.nmt.edu

Abstract

Parallel applications with deadline and security constraints are emerging in various areas like education, information technology, and business. However, conventional job schedulers for clusters generally do not take security requirements of real-time parallel applications into account when making allocation decisions. In this paper, we address the issue of allocating tasks of parallel applications on clusters subject to timing and security constraints in addition to precedence relationships. A task allocation scheme, or TAPADS (Task Allocation for Parallel Applications with Deadline and Security Constraints), is developed to find an optimal allocation that maximizes quality of security and the probability of meeting deadlines for parallel applications. In addition, we proposed mathematical models to describe a system framework, parallel applications with deadline and security constraints, and security overheads. Experimental results show that TAPADS significantly improves the performance of clusters in terms of quality of security and schedulability over three existing allocation schemes.

1. Introduction

In the last decade, clusters have become increasingly popular as powerful and cost-effective platforms for executing parallel applications [18]. Much of this trend can be attributed to rapid advances in processing power, network bandwidth, and storage capacity. As parallel applications in most cases require massive computing power, it becomes extremely important to take advantage of cluster computing platforms where nodes are interconnected through high-speed networks, e.g. Myrinet or fast Ethernet, to meet the needs of highly parallel applications [5][17].

There have been extensive studies in the literature on scheduling of real-time parallel applications on clusters [14][19]. Applications with timing constraints depend not only on results of computation, but also on time instants at which these results become available [12]. Examples of real-time parallel applications include aircraft control, radar for tracking missiles, medical electronics, and on-line transaction processing systems.

There is some work addressing the issue of applications with both deadline and security constraints [21]. In particular, sensitive data and processing in various real-time applications on clusters require special safeguard and protection against unauthorized access. However, conventional task allocation schemes designed for real-time systems are inappropriate in cases where parallel applications have both real-time and security requirements. This is because the traditional task allocation schemes were merely devised to guarantee timing constraints while possibly posing unacceptable security risks.

In this paper, we address the issue of task allocation in clusters for parallel applications with deadline, security, and task precedence constraints. A task allocation scheme, or TAPADS (Task Allocation for Parallel Applications with Deadline and Security Constraints), is developed to incorporate security and timing correctness into the process of making allocation decisions. TAPADS makes use of critical path analysis as well as security level refinement to maximize quality of security (measured by the probability of being risk free) and schedulability (measured by the probability of meeting task deadlines). We use an array of synthetic benchmarks to compare the proposed scheme with three baseline approaches. Empirical results show that TAPADS can guarantee timing constraints while achieving high security for parallel applications on clusters.

The rest of the paper is organized as follows. In the next section we summarize related work in the realm of real-time applications and computer security. Section 3 describes the model of system architecture and real-time tasks. In Section 4, we propose the TAPADS scheme for parallel applications running on clusters. Section 5 discusses security correctness of the allocation scheme. We present in Section 6 experimental results based on synthetic benchmarks. Section 7 concludes the paper with summary and future directions.

2. Related work

The issue of allocating/scheduling tasks in a cluster has been extensively addressed in the past both experimentally and theoretically [29]. Subramani *et al*. incorporated a buddy scheme for contiguous node allocation into a backfilling job scheduler for clusters [22]. Vallee *et al*. proposed a global scheduler architecture that can dynamically change scheduling policies while applications are running on clusters [24]. All the above schemes were designed for applications without deadline constraints.

There exist extensive studies on task allocating/scheduling for real-time applications [1]. In general, real-time task allocating/scheduling schemes fall into two categories: static (off-line) [15] and dynamic (on-line). Some algorithms were developed for real-time tasks that are independent of one other [23], whereas others were designed for tasks with precedence constraints [19]. Hou and Shin proposed a task allocation scheme to schedule periodic tasks with precedence constraints in distributed real-time systems [15]. He *et al*. studied the problem of dynamic scheduling of parallel real-time jobs executing on heterogeneous clusters [14]. All these approaches provide high schedulability for real-time systems. The above schemes were constructed for real-time applications with no security requirements, thereby ignoring security constraints needed to be stringently enforced for security-sensitive real-time systems.

Increasing attention has been drawn toward the problem of security in the context of clusters, because efficient and flexible security is a fundamental requirement for contemporary clusters. Apvrille and Pourzandi developed a new security policy language named distributed security policy, or DSP, for clusters [3]. Azzedin and Maheswaran integrated the notion of "trust" into resource management of a large-scale wide-area system [4]. The above security techniques were developed for non-real-time applications, and were unable to express and handle deadline constraints.

There has been some work incorporating security into a diversity of real-time applications. Ahmed and Vrbsky proposed a secure optimistic concurrency control protocol that can make trade-offs between security and real-time requirements [1]. Son *et al*. studied an approach to trading off quality of security to achieve required real-time performance [20]. Our work is in sharp contrast to the above approaches in the sense that their techniques were focused on concurrency control protocols whereas ours is intended to develop a real-time task allocation scheme, which can meet both timing and security constraints of parallel applications on clusters.

3. Mathematical models

We describe in this section mathematical models, which were built to represent a task allocation framework, security overheads, and parallel applications with security, precedence, and deadline constraints.

3.1 System model

A cluster in the most general form consists of a set $M = \{M_1, M_2, ..., M_m\}$ of nodes connected by a network. It is assumed that all nodes have an identical processing power. This assumption is reasonable in the sense that it can be relaxed by incorporating a conversion mechanism for relative heterogeneous processing capabilities. Each node communicates with other nodes through message passing, and the communication time between two tasks assigned to the same node is assumed to be negligible.

Note that the communication subsystem can support messages with time constraints, meaning that the worst-case link delay is predictable and bounded. Examples of such real-time communication subsystems can be found in the literature [26]. Additionally, the communication subsystem considered in our study provides full connectivity in a way that any two nodes are connected through either a physical link or a virtual link. This assumption is arguably reasonable for modern interconnection networks (e.g. Myrinet [5]) that are widely used in high-performance clusters. Myrinet networks provide pseudo-full connectivity, allowing simultaneous transfers between any pair of nodes.

3.2 Task model

3.2.1 Deadline and precedence constraints. Applications with dependent real-time tasks can be modeled by *Directed Acyclic Graphs (DAGs)*[19]. Throughout this paper, a parallel application is defined as a vector *(T, E, p)*, where $T = \{t_1, t_2,...,t_n\}$ represents a set of non-preemptive real-time tasks, E is a set of weighted and directed edges used to represent communication among tasks, e.g., $(t_i, t_j) \in E$ is a message transmitted from task t_i to t_j, and p_i is the period. Precedence constrains of the parallel application is represented by all the edges in E. Communication time for sending a message $(t_i, t_j) \in E$ from task t_i on m_k to task t_j on m_a is determined by e_{ij}/b_{ka} where e_{ij} is the data size and b_{ka} is the network bandwidth between m_k and m_a.

This paper is focused on the issue of allocating periodic tasks that are invoked at fixed time intervals and constitutes the base load of a cluster. A task is characterized by three parameters, e.g., $t_i = (e_i, l_i, S_i)$, where e_i is the execution, l_i denotes the amount of data (measured in KB) to be protected, and S_i is a vector of security requirements (see section 3.2.2.). Note that e_i can be estimated by code profiling and statistical prediction [7].

A parallel application running on the cluster generates a sequence of application instances $J_i^0, J_i^1, J_i^2 ...$, where J_i^j must be finished before J_i^{j+1} can start executing. Note that there is a constant interval between two consecutive application instances. The deadline of J_i^j is the arrival time of the next task instance. Although the arrival time of a task instance is not explicitly specified in the model, the arrival time can be determined when the task instance is released dynamically during the system execution. Specifically, if the initial release time of task J_i is r, the arrival time of the jth instance is $r + j \times p_i$, and the task instance has to be completed before $r + (j+1) \times p_i$. An application meets its deadline if all its instances meet their deadlines. A parallel application has a feasible schedule if for all $t \in T$, the deadline and security constraints are satisfied.

It has been proved that there exists a feasible schedule for a set of periodic tasks if and only if there is a feasible schedule for the *planning cycle* of the tasks [15]. Note that the planning cycle is the least common multiple of all the tasks' periods. Thus, the behavior of the set of periodic tasks can be effectively analyzed within the planning cycle.

3.2.2 Security constraints. A collection of security services required by task t_i is specified as $S_i = (S_i^1, S_i^2, \ldots, S_i^q)$, where S_i^j represents the required security level range of the jth security service. The TAPADS allocation scheme aims at determining the most appropriate point s_i in space S_i, e.g., $s_i = (s_i^1, s_i^2, \ldots, s_i^q)$, where $s_i^j \in S_i^j$, $1 \le j \le q$.

The proposed TAPADS scheme measures security benefits gained by parallel applications. To implement this basic functionality, we quantitatively model the security benefit of the jth task in T as a security level function denoted by $SL: S_i \to \Re$, where $\Re$ is the set of positive real numbers:

$$SL(s_i) = \sum_{k=1}^{q} w_i^j s_i^j \text{ where } s_i = \left(s_i^1, s_i^2, ..., s_i^q\right),$$
$$0 \le w_i^j \le 1, \text{ and } \sum_{j=1}^{q} w_i^j = 1. \quad (1)$$

Users can specify the weight w_i^j to reflect relative priorities given to the jth security service. The security benefit of a task set is computed as the summation of the security levels of all the tasks. Thus,

$$SL(T) = \sum_{i=1}^{n} SL(s_i), \text{where } s_i = \left(s_i^1, s_i^2, ..., s_i^q\right). \quad (2)$$

Given a task set T, We can obtain the following non-linear optimization problem:

$$\text{maximize } SL(T) = \sum_{i=1}^{n}\sum_{k=1}^{q} w_i^j s_i^j,$$
$$\text{subject to } \min(S_i^j) \le s_i^j \le \max(S_i^j), \quad (3)$$

where $\min(S_i^j)$ and $\max(S_i^j)$ are the minimum and maximum security requirements of task t_i.

An array of security services required by massage $(t_i, t_j) \in E$ is specified as $\hat{S}_{ij} = (\hat{S}_{ij}^1, \hat{S}_{ij}^2, ..., \hat{S}_{ij}^p)$, where $\hat{S}_{ij}^k$ denotes the required security level range of the kth security service. The most appropriate point $\hat{s}_{ij}$ in space $\hat{S}_{ij}$ has to be calculated, e.g. $\hat{s}_{ij} = (\hat{s}_{ij}^1, \hat{s}_{ij}^2, ..., \hat{s}_{ij}^p)$, where $\hat{s}_{ij}^k \in \hat{S}_{ij}^k$, $1 \le k \le p$. We model the security benefit of the message as the following function:

$$SL(\hat{s}_{ij}) = \sum_{k=1}^{p} \hat{w}_{ij}^k \hat{s}_{ij}^k, \text{where } \hat{s}_{ij} = (\hat{s}_{ij}^1, \hat{s}_{ij}^2, ..., \hat{s}_{ij}^p),$$
$$0 \le \hat{w}_{ij}^k \le 1, \text{ and } \sum_{j=1}^{p} \hat{w}_{ij}^k = 1. \quad (4)$$

Weight $\hat{w}_{ij}^k$ reflects priorities of the kth required security services for the message. The security benefit of a message set is calculated as the sum of the security levels of all the messages. That is,

$$SL(E) = \sum_{(t_i,t_j)\in E} SL(\hat{s}_{ij}),$$

where $\hat{s}_{ij} = (\hat{s}_{ij}^1, \hat{s}_{ij}^2, ..., \hat{s}_{ij}^p)$. (5)

The optimal security benefit of the message set can be computed as follows:

$$\text{maximize } SL(E) = \sum_{(t_i,t_j)\in E} \sum_{k=1}^{p} \hat{w}_{ij}^k \hat{s}_{ij}^k,$$

$$\text{subject to } \min(\hat{S}_{ij}^k) \le \hat{s}_{ij}^k \le \max(\hat{S}_{ij}^k), \quad (6)$$

where $\min(\hat{S}_{ij}^k)$ and $\max(\hat{S}_{ij}^k)$ are the minimum and maximum security requirements of the message.

Now we can define an optimization problem formulation to compute the optimal security benefit of a parallel applications, subject to certain timing and security constraints:

$$\text{Maximize } SV = SL(T) + SL(E). \quad (7)$$

Substituting Equations (3) and (6) into (7) yields the following security value objective function. The task allocation problem can be formally defined as follows: given a cluster $M = \{M_1, M_2, ..., M_n\}$ and a parallel application $\{T, E\}$, find an allocation each task and message, such that the following total security level of the application on the cluster is maximized.

$$SV = \sum_{i=1}^{n} \sum_{k=1}^{q} w_i^k s_i^k + \sum_{(t_i,t_j)\in E} \sum_{k=1}^{p} \hat{w}_{ij}^k \hat{s}_{ij}^k, \quad (8)$$

3.3 Security overhead model

Recently Irvine and Levin proposed a security overhead framework, which can be used for a variety of purposes [16]. However, security overhead model for each security services in the context of cluster computing remains an open issue. In this section we proposed a model to measure security overheads incurred by parallel applications with security constraints. Without loss of generality, we consider three security services widely deployed in clusters, namely, confidentiality, authentication, and integrity check.

Confidentiality is used to encrypt applications and data so that a third party is unable to discover users' private information. We assume that each node has nine optional encryption algorithms [27]. Each cryptographic algorithm is assigned a security level based on its performance. Integrity services make it possible to ensure that no one can modify or tamper applications while they are executing on clusters without being detected. This can be accomplished by using a variety of hash functions [6]. Ten commonly used hash functions and their performance (evaluated on a 90 MHz Pentium machine) are shown in [28]. Tasks must be submitted from authenticated users and, thus, authentication services are deployed to authenticate users who wish to access the cluster [8][10][13]. Table 3 in [27] lists three authentication techniques: weak authentication using HMAC-MD5; acceptable authentication using HMAC-SHA-1, and fair authentication using CBC-MAC-AES.

Now we can derive security overhead, which is the sum of the three items above. Suppose task t_i requires the three security services above, which are provided in sequential order. Let s_i^j and $c_i^j(s_i^j)$ be the security level and overhead of the jth security service, the security overhead c_i experienced by t_i, can be computed using Equation (9).

$$c_i = \sum_{j\in\{a,e,g\}} c_i^j(s_i^j), \text{ where } s_i^j \in S_i^j. \quad (9)$$

Detailed information pertinent to security overhead model can be found in [27][28].

4. Task allocation algorithm

This section presents a task allocation algorithm, or TAPADS, for parallel applications with deadline and security constraints on clusters. Let X be an m by n binary matrix corresponding to an allocation, in which n tasks are assigned to m nodes in the cluster. Element x_{ij} equals 1 if and only if t_i has been allocated to node m_j; otherwise $x_{ij} = 0$. Thus, we have $M(t_i) = j \Leftrightarrow x_{ij} = 1$, where $M(t_i)$ indicates the node to which t_i is allocated. The TAPADS algorithm strives to maximize $P_{OSP}(X) = P_{SC}(X) \cdot P_{SD}(X)$ subject to $\sum_{j=1}^{m} x_{ij} = 1 \quad i \in [1,n]$, where $P_{OSP}(X)$ is the overall system performance, $P_{SC}(X)$ is the probability that all tasks are executed without any risk of being attacked and all messages are risk-free during the course of message transmissions, and $P_{SD}(X)$ is a fraction of total submitted jobs that are schedulable. Note that $P_{SC}(X)$ will be derived in Section 5.

1. Sort and renumber tasks so that if $(t_i, t_j) \in E$ then $i < j$;
2. Compute the critical path of the task graph;
3. Calculate the tentative finish time, $f = \sum_{t_i \in \text{critical path}} (e_i + c_i^{\min})$;
4. Allocate and schedule all t_i in the critical path subject to minimal security requirements;
5. Allocate and schedule all $t_i \notin$ critical path subject to precedence and minimal security constraints;
6. Obtain slack time, $slk = d - f$, where d is the deadline;
7. **while** (slack time ≥ 0) **do**
   - 7.1 select i' and j' subject to
     $$w_{i'}^{j'} \Delta s_{i'}^{j'} \Big/ \left(c_{i'}^{j'}(s_{i'}^{j'} + \Delta s_{i'}^{j'}) - c_{i'}^{j'}(s_{i'}^{j'})\right) = \max_{1 \le i \le n, 1 \le j \le q} \left\{ w_i^j \Delta s_i^j \Big/ \left(c_i^j(s_i^j + \Delta s_i^j) - c_i^j(s_i^j)\right)\right\}$$
   - 7.2 select a' and b' subject to
     $$\hat{w}_{a'b'}^{k'} \Delta \hat{s}_{a'b'}^{k'} \Big/ \left(\hat{c}_{a'b'}^{k'}(\hat{s}_{a'b'}^{k'} + \Delta \hat{s}_{a'b'}^{k'}) - \hat{c}_{a'b'}^{k'}(\hat{s}_{a'b'}^{k'})\right) = \max_{\substack{(t_a, t_b) \in E \\ 1 \le k \le p}} \left\{ \hat{w}_{ab}^{k} \Delta \hat{s}_{ab}^{k} \Big/ \left(\hat{c}_{ab}^{k}(\hat{s}_{ab}^{k} + \Delta \hat{s}_{ab}^{k}) - \hat{c}_{ab}^{k}(\hat{s}_{ab}^{k})\right)\right\}$$
   - 7.3 **if** $w_{i'}^{j'} \Delta s_{i'}^{j'} \Big/ \left(c_{i'}^{j'}(s_{i'}^{j'} + \Delta s_{i'}^{j'}) - c_{i'}^{j'}(s_{i'}^{j'})\right) > \hat{w}_{a'b'}^{k'} \Delta \hat{s}_{a'b'}^{k'} \Big/ \left(\hat{c}_{a'b'}^{k'}(\hat{s}_{a'b'}^{k'} + \Delta \hat{s}_{a'b'}^{k'}) - \hat{c}_{a'b'}^{k'}(\hat{s}_{a'b'}^{k'})\right)$ **then**
     increase security level $s_{i'}^{j'} \leftarrow s_{i'}^{j'} + \Delta s_{i'}^{j'}$ if $s_{i'}^{j'} < \max\{S_{i'}^{j'_l}\}$;
   - 7.4 **else** increase security level $\hat{s}_{a'b'}^{k'} \leftarrow \hat{s}_{a'b'}^{k'} + \Delta \hat{s}_{a'b'}^{k'}$ if $\hat{s}_{a'b'}^{k'} < \max\{\hat{S}_{a'b'}^{k'_l}\}$;
   - 7.5 update task schedule, e.g., start times of tasks and messages;
   - 7.6 update slack time based on the increased security level;

   end while

Figure 1. The TAPADS task allocation algorithm

The TAPADS algorithm is outlined in Figure 1. TAPADS aims at achieving high quality of security under two conditions: (1) increasing security levels will not result in missing deadlines; and (2) precedence constraints are satisfied. In an effort to meet both deadline and precedence constraints, TAPADS assigns the tasks to each node in a way to maximize security measured as $P_{SC}(X)$. Thus, TAPADS is capable of maintaining a high schedulability measured as $P_{SD}(X)$.

Before optimizing the security level of each task and message of a job, TAPADS makes the best effort to satisfy the deadline and precedence constraints. This can be accomplished by calculating the earliest start time and the minimal security overhead of each task and message in Steps 4 and 5. If the deadline can be guaranteed provided that the minimal security requirements are met, the slack time of the initial task allocation can be obtained by Step 6.

To efficiently improve quality of security of the job, in Step 7 TAPADS chooses the most appropriate task or message in which the security level will be increased. Specifically, it is desirable to give higher priorities to security services with higher weights and lower security overhead. Hence, we define the following two benefit-cost ratio functions, e.g., θ_i^j and $\hat{\theta}_{ij}^k$, which measure the increase of security level by unit security overhead.

$$\theta_i^j = w_i^j \Delta s_i^j \Big/ \left(c_i^j(s_i^j + \Delta s_i^j) - c_i^j(s_i^j)\right), \text{ for the } j\text{th service of task } i, \quad (10)$$

$$\hat{\theta}_{ij}^k = \hat{w}_{ij}^k \Delta \hat{s}_{ij}^k \Big/ \left(\hat{c}_{ij}^k(\hat{s}_{ij}^k + \Delta \hat{s}_{ij}^k) - \hat{c}_{ij}^k(\hat{s}_{ij}^k)\right), \text{ for the } k\text{th service of message } (t_i, t_j), \quad (11)$$

where the numerators represent the weighted increase in the security level, whereas the denominators indicate the corresponding increase in security overhead.

After performing Steps 7.1 and 7.2, TAPADS identifies the best candidate in $V \cup E$ that has the highest benefit-cost ratio. Formally, the best candidate is chosen based on the following expression,

$$\begin{cases} \theta_{i'}^{j'} = \max\limits_{1 \le i \le n, 1 \le j \le q} \{\theta_i^j\}, \text{if } \max\limits_{1 \le i \le n, 1 \le j \le q} \{\theta_i^j\} \ge \max\limits_{(t_i, t_j) \in E, 1 \le k \le p} \{\hat{\theta}_{ij}^k\}, \\ \theta_{i'j'}^{k'} = \max\left\{ \max\limits_{(t_i, t_j) \in E, 1 \le k \le p} (\hat{\theta}_{ij}^k) \right\}, \quad \text{otherwise.} \end{cases} \quad (12)$$

To yield a maximized security level of the job, Steps 7.3 and 7.4 are responsible for increasing security levels of more important services at the minimal cost. Thus, the slack time is distributed on a task or message with the highest benefit-cost ratio.

Step 7.5 updates the schedule in accordance with the increased security level, because start times of other tasks and messages are dependent of how the slack time is distributed. Finally, step 7.6 updates the slack time based on the increased security level.

The time complexity of the *SAREG* scheduling algorithm is given as follows.

Theorem 1. The time complexity of TAPADS is *O(k(q|V|+p|E|))*, where *k* is the number of times Step 7 is repeated, *q* is the number of security services for computation, *p* is the number of security service for communication.

Proof. The time complexity of allocating and scheduling tasks subject to precedence and minimal security constraints is *O(|V|+|E|)* (Steps 1-6). To effectively boost security levels of tasks and messages under the constraints (Steps 7.3-7.4), it takes time *O(|V|+|E|)* to select the most appropriate task or message as a candidate whose quality of security will be improved. The time complexity of step 7 becomes *O(k(q|V|+p|E|))*. Thus, the time complexity of TAPADS is: *O(|V|+|E|)* + *O(k(q|V|+p|E|))* = *O(k(q|V|+p|E|))*.

k cannot be very big numbers in practice, because *k* in many cases is much smaller than |*V*|+|*E*|. Therefore, the time complexity of TAPADS is reasonably low based on the expression above.

5. Evaluation of security correctness

To evaluate quality of security for parallel applications, we derive in this section the probability $P_{SC}(X)$ that all tasks and messages remain risk-free during the course of execution.

The quality of security of a task t_i with respect to the *j*th security service is calculated as $\exp(-\lambda_i^j e_i)$, where λ_i^j is t_i's risk rate of the *j*th security service and e_i is the execution time. The risk rate is expressed as:

$$\lambda_i^j = 1 - \exp\left(-\alpha(1 - s_i^j)\right). \quad (13)$$

Note that this risk rate model assumes that risk rates are a function of security levels, and the distribution of risk-count for any fixed time interval is approximated using a *Poisson* probability distribution. The risk rate model is just for illustration purpose only. Thus, the model can be replaced by any risk rate model with a reasonable parameter α.

The quality of security of a task t_i can be obtained below by considering all security services provided to the task. Consequently, we have:

$$\prod_{j=1}^{q} \exp(-\lambda_i^j e_i) = \exp\left(-e_i \sum_{j=1}^{q} \lambda_i^j\right). \quad (14)$$

Given an allocation *X*, the probability that all tasks are free from being attacked during the execution of the tasks is computed based on Equation (14), thus,

$$P_C(X) = \prod_{i=1}^{n} \exp\left(-e_i \sum_{j=1}^{q} \lambda_i^j\right). \quad (15)$$

By substituting the risk rate model into Equation (15), we finally obtain $P_C(X)$ as shown below:

$$P_C(X) = \prod_{i=1}^{n} \exp\left\{-e_i \sum_{j}^{q} \left[1 - \exp\left(-\alpha(1 - s_i^j)\right)\right]\right\}. \quad (16)$$

Likewise, for the *k*th security service available for a link between M_i and M_j, the quality of security of the link during the time interval *t* is $\exp(-\hat{\lambda}_{ij}^k t)$, where $\hat{\lambda}_{ij}^k$ denotes the risk rate of the *k*th security service. The risk rate is expressed as the following function of the corresponding security level.

$$\hat{\lambda}_{ij}^k = 1 - \exp\left(-\beta(1 - s_{ij}^k)\right). \quad (17)$$

The quality of security of a message $(t_a, t_b) \in E$ is calculated by taking all security services provided to the message into account. Thus,

$$\prod_{k=1}^{p} \exp(-\hat{\lambda}_{ij}^k \frac{d_{ab}}{B_{ij}}) = \exp\left(-\frac{d_{ab}}{B_{ij}} \sum_{k=1}^{p} \hat{\lambda}_{ij}^k\right),$$

where $x_{ai} = 1, x_{bj} = 1$. (18)

Given an allocation *X*, the probability that all messages allocated to the link between M_i and M_j are risk-free is computed as the product of the quality of security of all the messages. Then, we have:

$$P_{ij}(X) = \prod_{a=1}^{n} \prod_{b=1, b\neq a}^{n} \exp\left[x_{ai} x_{bj} \left(-\frac{d_{ab}}{B_{ij}} \sum_{k=1}^{p} \hat{\lambda}_{ij}^k\right)\right]. \quad (19)$$

We now obtain $P_{ij}(X)$ by substituting the risk rate model into Equation (19),

$$P_{ij}(X) = \prod_{a=1}^{n} \prod_{b=1, b\neq a}^{n} \exp\left\{x_{ai} x_{bj} \left[-\frac{d_{ab}}{B_{ij}} \sum_{k=1}^{p} \left(1 - \exp\left(-\beta(1 - s_{ij}^k)\right)\right)\right]\right\}. \quad (20)$$

Let $P_L(X)$ be the quality of security experienced by all links under allocation X, and $P_L(X)$ can be written as the following equation:

$$P_L(X) = \prod_{i=1}^{m} \prod_{j=1, j\neq i}^{m} P_{ij}(X). \quad (21)$$

Finally, we the probability $P_{SC}(X)$ can be calculated as follows, where $P_C(X)$ and $P_L(X)$ are obtained from Equations (16) and (21).

$$P_{SC}(X) = P_C(X)P_L(X). \quad (22)$$

service v_l, the following equation is always held: $s_i^{v_j} = \min\{S_i^{v_j}\}$.

(2) *LISTMAX*: The scheduler chooses the highest security level for each security requirement posed by each task within a parallel job. This fact can be formally represented by the following expression: $\forall T_i, v_j \in \{a, e, g\} : s_i^{v_j} = \max\{S_i^{v_j}\}$.

(3) *LISTRND*: Unlike the above two baseline algorithms, LISTRND randomly picks a value within the security level range of each service required by a task. The following expression is held $\forall T_i, v_j \in \{a, e, g\} : s_i^{v_j} = \text{random}\{S_i^{v_j}\}$.

Table 1. Characteristics of system parameters

Parameter	Value
CPU Speed	100 million instructions/second or MIPS
Network bandwidth	100 Mbps
Task execution time	(min, top, max)=(1,5, 10), (10,20,40), (40,80,160), (160,320,640) seconds
Number of nodes	32
Deadlines	(100, 200, 300, 400, 500, 600) seconds
Deadline ranges	([100, 200], [200, 300], [300, 400], [400, 500]) seconds
Out degrees	(25, 50, 75, 100)
Size of data to be secured	(min, top, max)=(0.02, 0.1, 0.5), (0.2, 1, 5), (1, 5, 10), (10, 20, 30) MB
Required security services	Encryption, Integrity, and Authentication
Weight of security services	0.2 (authentication), 0.5 (encryption), 0.3 (integrity)

6. Performance evaluation

Now we are in a position to evaluate the effectiveness of TAPADS using extensive simulations. To demonstrate the strength of TAPADS, we compare it with list scheduling scheme, which is a well-known scheduler for parallel applications. To make the comparison fair, we slightly modified it into three variants: LISTMIN, LISTMAX, and LISTRND. The variants can meet parallel applications' security requirements in a heuristic way. Although these algorithms are intended to schedule real-time tasks with security requirements, they make no effort to optimize quality of security. The three baseline algorithms are briefly described below.

(1) *LISTMIN*: The scheduler intentionally selects the lowest security level of each security service required by each task of a parallel job. Thus, given task T_i and

6.1 Simulator and simulation parameters

Before presenting empirical results, we present the simulation model as follows. Table 1 summarizes the configuration parameters of simulated clusters used in our experiments. The parameters of nodes in the clusters are chosen to resemble real-world workstations like Sun SPARC-20 and Sun Ultra 10.

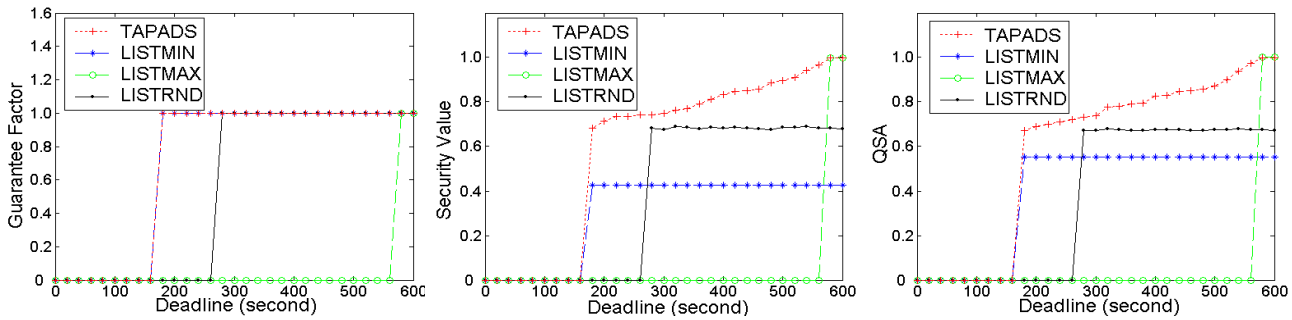
All synthetic parallel jobs used from Section 6.2 to Section 6.3 were created by TGFF [9], a randomized task graph generator. Although deadlines, size of data to be secured, numbers of out degree, task execution time are synthetically generated, we examined impacts of these important workload parameters on system performance by controlling the parameters. The performance metrics by which we evaluate system performance include:

- *Security Value* (see Equation 8).

• *Schedulability*: a fraction of total submitted jobs that are schedulable.
• *Quality of security (QSA)*: quality of security for applications (see Equation 22).
• *Guarantee factor*: it is zero if a job's deadline cannot be met. Otherwise, it is one.

6.2 Overall Performance Comparisons

The goal of this experiment is to compare the proposed *TAPADS* algorithm against the other three baseline schemes, and to understand the sensitivity of *TAPADS* to deadlines. We tested task graphs with 433 tasks with precedence constraints.



(a) Guarantee Factor (b) Security Value (c) Quality of Security

Figure 2. Performance impact of deadline

Figure 2 shows the simulation results for these four algorithms on a cluster with 32 nodes. We statically computed the minimal job completion time defined as a job's completion time when the minimal security requirements of each task and message are met. Similarly, we can calculate the random job completion time and the maximal job completion time. For the tested parallel applications, the minimal, random, and maximal job completion times are 170, 206, and 575 seconds, respectively. We observe from Figure 2 (a) that TAPADS and LISTMIN exhibit same performance in terms of guarantee factor, whereas TAPADS noticeably outperforms the LISTMAX and LISTRND algorithms. Once deadline is longer than 170 seconds, both TAPADS and LISTMIN have capability of generating feasible schedules, whereas no feasible schedule can be yielded by LISTRND and LISTMAX until deadlines are longer than 206 and 575 seconds, respectively. This is because high security overhead results in long completion times. We attribute the performance improvement of TAPADS over LISTMAX and LISTRND to the fact that TAPADS boosts the security levels of tasks and messages under the condition that timing constraints are guaranteed and, therefore, TAPADS maintains the same guarantee factor performance as that of LISTMIN. In contrast, the LISTMAX and LISTRND policies improve the quality of security at the cost of missing deadlines.

Figure 2 (b) plots security values of the four algorithms when the deadline is increased from 0 to 600 seconds. It reveals that TAPADS consistently performs better than LISTMIN and LISTRND. In particular, TAPADS achieves improvement on averages of 97.7% and 25.0%, respectively. This is because LISTMIN and LISTRND do not utilize slack times to increase security levels of tasks and messages. In the case where the deadlines are longer than the maximal job completion time, TAPADS eventually degrades to LISTMAX. Importantly, TAPADS substantially outperforms LISTMAX when the deadlines are tight. The results clearly indicate that clusters can gain more performance benefits from our TAPADS approach under the circumstance that real-time applications have relatively tight deadlines.

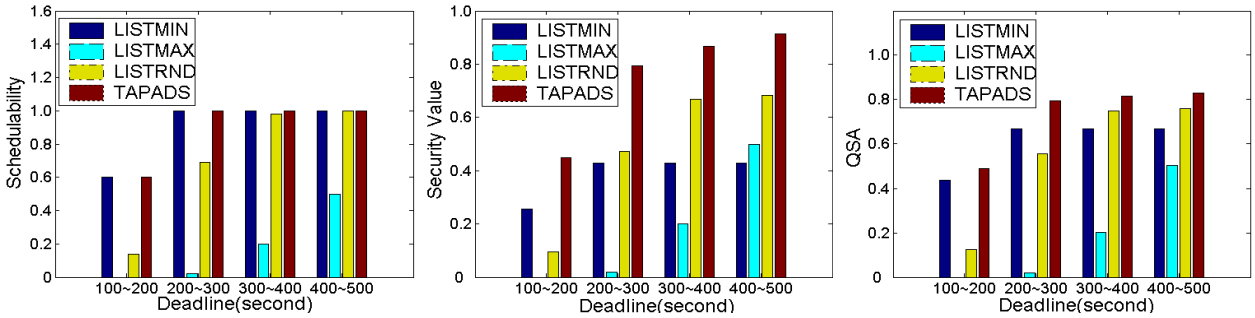
Improvements in the quality of security achieved by TAPADS are plotted in Figure 2 (c). Compared with LISTMIN and LISTRND, TAPADS achieves improvement on averages of 54.5% and 25.7%, respectively. The first observation deduced from Figure 2 (c) is that the quality of security of TAPADS increases with the deadline. This is because quality of security is partially derived from SV (see Equations 16 and 21), which becomes higher when the deadlines are looser. A second observation is that the performance improvement of TAPADS in terms of quality of security is not as pronounced as the performance improvement in terms of security value compared with LISTMIN algorithm. This can be explained by the negative natural exponential function (see Equations 13 and 17), which smoothes the security value differences between LISTMIN and TAPADS.

6.3 Adaptability

We conducted four groups of experiments to test the performance of TAPADS using 1000 diverse task

graphs. The smallest task graph has 54 tasks, and the largest task graph consists of 543 tasks. We assume that the number of nodes in the cluster is 32. For each group test, we set a deadline range from which a deadline is randomly selected for an incoming parallel job. The four deadline ranges for the four group experiments are [100, 200], [200, 300], [300, 400] and [400, 500], respectively.

framework, parallel applications with deadline and security constraints, and security overheads. These models are used by the TAPADS scheme to measure security overheads caused by an array of security services, including encryption, authentication, integrity check, etc. To quantitatively evaluate the effectiveness and practicality of the proposed TAPADS scheme, we conducted extensive experiments using synthetic



(a) Schedulability (b) Security Value (c) Quality of Security

Figure 3. Group experiment for deadline

Figure 3 shows the performance impacts of the deadlines. We observe from Figure 3 (a) that the TAPADS and LISTMIN deliver the best performance in schedulability under all four cases. Figure 3 (b) demonstrates that TAPADS consistently outperforms the others in terms of security value. Interestingly, the improvement of TAPADS in security over LISTMAX is significant, although LISTMAX attempts to meet the maximal security requirements. This is mainly due to the low schedulability of LISTMAX (e.g., LISTMAX merely provides feasible schedules for 50% parallel applications). Figure 3 (c) clearly shows that TAPADS noticeably delivers the best quality of security.

7. Conclusions

In this paper, we address the issue of allocating tasks of parallel applications on clusters subject to timing and security constraints in addition to precedence relationships. TAPADS (Task Allocation for Parallel Applications with Deadline and Security Constraints), a task allocation scheme for parallel application with deadline and security constraints, is developed to generate optimal allocations that maximize quality of security and the probability of meeting deadlines for parallel applications running on clusters. The proposed TAPADS scheme factors in security and timing correctness in a way that the probabilities of being risk free and meeting deadlines are used as the performance objectives for clusters. To facilitate the presentation of TAPADS, we also proposed mathematical models to describe a system benchmarks. Our experimental results show that TAPADS significantly improves the performance of clusters in terms of quality of security and schedulability over three existing allocation schemes. Compared with LISTMIN and LISTRND, TAPADS achieves improvement in security value on averages of 97.7% and 25.0%, respectively. The improvement of TAPADS in quality of security over LISTMIN and LISTRND are 54.5% and 25.7%, respectively. TAPADS improves the schedulability performance over LISTMAX by an average of 400%.

Future studies in this research can be performed in the following directions. First, we will extend our security overhead model to multi-dimensional computing resources. Second, besides the three security services discussed, we plan to take into account of authorization and auditing services. Last, we intend to enable the TAPADS scheme to deal with heterogeneous clusters, where different nodes have various computing capacities and resources.

Acknowledgements

This work was partially supported by a research grant from the Intel Corporation and a start-up research fund (103295) from the Research and Economic Development Office of the New Mexico Institute of Mining and Technology. The authors are grateful to Ronald Tafoya and Noel Paz of the Intel Corporation for their stimulating discussions on the approach and development of this work. The authors also extend their thanks to Andrew Sung of the Computer Science

Department at the New Mexico Tech for his contributions during the early stages of this study.

References

[1] 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.

[2] Q. Ahmed and S. Vrbsky, "Maintaining security in firm real-time database systems," *Proc. 14th Ann. Computer Security Application Conf.*, 1998.

[3] A. Apvrille and M. Pourzandi, "XML Distributed Security Policy for Clusters," *Computers & Security Journal*, Elsevier, Vol.23, No.8, pp. 649-658, Dec. 2004.

[4] F. Azzedin, M. Maheswaran, "Towards trust-aware resource management in grid computing systems," *Proc. 2nd IEEE/ACM Int'l Symp. Cluster Computing and the Grid*, May 2002.

[5] N.J. Boden, D. Cohen, and W.K. Su., "Myrinet: A Gigabit-per-second Local Area Network," IEEE Micro, Vo. 15, No. 1, February 1995.

[6] A. Bosselaers, R. Govaerts and J. Vandewalle, "Fast hashing on the Pentium," *Proc. Advances in Cryptology*, LNCS 1109, pp. 298-312, Springer-Verlag, 1996.

[7] T. D. Braun *et al*., "A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems," *Proc. Workshop on Heterogeneous Computing,* pp.15-29, Apr. 1999.

[8] J. Deepakumara, H.M. Heys, and R. Venkatesan, "Performance comparison of message authentication code (MAC) algorithms for Internet protocol security (IPSEC)," *Proc. Newfoundland Electrical and Computer Engineering Conf.*, St. John's, Newfoundland, Nov. 2003.

[9] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF:Task graphs for free," *Proc. Int. Workshop. Hard-ware/Software Codesign*, pp. 97-101, Mar. 1998.

[10] O. Elkeelany, M. Matalgah, K. Sheikh, M. Thaker, G. Chaudhry, D. Medhi, J. Qaddouri, "Performance analysis of IPSec protocol: encryption and authentication," *Proc. IEEE Int'l Conf. Communications*, pp. 1164-1168, New York, NY, April-May 2002.

[11] R. F. Freund, *et al*. "Scheduling resources in multi-user, heterogeneous computing environments with SmartNet," *Proc. Heterogeneous Computing Workshop*, pp.184-199, March 1998.

[12] W. A. Halang, *et al*., "Measuring the performance of real-time systems," *Int'l Journal of Time-Critical Computing Systems*, 18, pp. 59-68, 2000.

[13] A. Harbitter and D. A. Menasce, "The performance of public key enabled Kerberos authentication in mobile computing applications," *Proc. ACM Conf. Computer and Comm. Security*, pp. 78-85, 2001.

[14] L. He, A. Jatvis, and D. P. Spooner, "Dynamic scheduling of parallel real-time jobs by modelling spare capabilities in heterogeneous clusters," *Proc. Int'l Conf. Cluster Computing*, pp. 2-10, Dec. 2003.

[15] 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, Dec. 1997.

[16] C. Irvine and T. Levin, "Towards a taxonomy and costing method for security services," *Proc. 15th Annual Computer Security Applications Conference*, 1999.

[17] X. Qin, H. Jiang, Y. Zhu, and D. R. Swanson, "Towards Load Balancing Support for I/O-Intensive Parallel Jobs in a Cluster of Workstations," *Proc. IEEE Int'l Conf. Cluster Computing*, pp.100-107, 2003.

[18] X. Qin, "Improving Network Performance through Task Duplication for Parallel Applications on Clusters," *Proc. 24th IEEE Int'l Performance, Computing, and Communications Conference,* pp.35-42, Phoenix, Arizona, April 2005.

[19] X. Qin, H. Jiang, D. R. Swanson, "An Efficient Fault-tolerant Scheduling Algorithm for Real-time Tasks with Precedence Constraints in Heterogeneous Systems," *Proc. 31st Int'l Conf. Parallel Processing,* pp.360-368. Aug. 2002.

[20] S. H. Son, R. Zimmerman, and J. Hansson, "An adaptable security manager for real-time transactions," *Proc. 12th Euromicro Conf. Real-Time Systems*, pp. 63 – 70, June 2000.

[21] 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, 2000.

[22] V. Subramani, V., R. Kettimuthu, S. Srinivasan, J. Johnston, and P. Sadayappan, "Selective buddy allocation for scheduling parallel jobs on clusters," *Proc. IEEE Int'l Conf. Cluster Computing*, pp. 107 – 116, Sept. 2002.

[23] 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.

[24] G. Vallee, C. Morin, J.-Y. Berthou, and L. Rilling, "A new approach to configurable dynamic scheduling in clusters based on single system image technologies," *Proc. Int'l Symp. Parallel and Distributed Processing*, April 2003.

[25] C.M. Woodside and G.G. Monforton, "Fast Allocation of Processes in Distributed and Parallel Systems", *IEEE Trans. on Parallel and Distributed Systems*, Vol. 4, No. 2, pp. 164-174, Feb. 1993.

[26] Q. Zheng and K.G. Shin, "On the Ability of Establishing Real-Time Channels in Point-to-Point Packet Switched Network," IEEE Trans. Comm., Vol. 42, 1994.

[27] T. Xie, X. Qin, "Enhancing Security of Real-Time Applications on Grids through Dynamic Scheduling," *Proc. the 11th Workshop on Job Scheduling Strategies for Parallel Processing,* pp.146-158, Cambridge, MA, USA, June, 2005.

[28] T. Xie, X. Qin, A. Sung, "SAREC: A Security-Aware Scheduling Strategy for Real-Time Applications on Clusters," *Proc. the 34th Int'l Conf. Parallel Processing*, PP. 5-12, Norway, June, 2005.

[29] X. Zhou, Y. Cai, C. Chow, and M. Augusteijn, "Two-Tier Resource Allocation for Slowdown Differentiation on Server Clusters," *Proc. the 34th Int'l Conf. Parallel Processing*, PP. 5-12, Norway, June, 2005.