A Security-Oriented Task Scheduler for Heterogeneous Distributed Systems

Tao Xie[1] and Xiao Qin[2]

[1] Department of Computer Science, San Diego State University,
San Diego, CA 92182, USA
xie@cs.sdsu.edu
[2] Department of Computer Science, New Mexico Institute of Mining and Technology,
Socorro, NM 87801, USA
xqin@cs.nmt.edu

Abstract. High quality of security is increasingly critical for applications running on heterogeneous distributed systems, where processors operate at different speeds and communication channels have different bandwidths. Although there are a few scheduling algorithms in the literature for heterogeneous distributed systems, they generally do not take into account of security requirements of applications. In this paper, we propose a novel heuristic scheduling algorithm, or SATS, which is conducive to improving security of heterogeneous distributed systems. First, we formalize a concept of security heterogeneity for our scheduling model in the context of distributed systems. Next, we devise the SATS algorithm aiming at scheduling tasks to maximize the probability that all tasks are executed without any risk of being attacked. Empirical results demonstrate that with respect to security and performance, the proposed scheduling algorithm outperforms existing approaches for heterogeneous distributed systems.

Keywords: Security heterogeneity, heterogeneous distributed system, scheduling, degree of security deficiency, risk-free probability.

1 Introduction

Over the last decade, heterogeneous distributed systems have been emerging as popular computing platforms for computationally intensive applications with diverse computing needs [6]. To date they have been applied to security sensitive applications, such as banking systems and digital government, which require new approaches to security. Inherently, distributed systems are more vulnerable to threats than centralized systems, since it is difficult to control processing activities of the distributed systems and information can be accessed over networks. A variety of techniques like authentication [8] and access control [12] are widely used to secure distributed systems. Although these techniques can be applied to distributed systems, the conventional security techniques lack the ability to express heterogeneity in security services. Our study is intended to introduce a concept of security heterogeneity, which provides a means of measuring overhead incurred by security services in the context of heterogeneous distributed systems.

Y. Roberts et al. (Eds.): HiPC 2006, LNCS 4297, pp. 35 – 46, 2006.

Scheduling algorithms play a key role in obtaining high performance in parallel and distributed systems [10][18]. The objective of scheduling algorithms is to map tasks onto sites and order their execution in a way to optimize overall performance. In this work we consider the issue of dynamic task scheduling. Nowadays, a wide variety of scheduling algorithms for distributed systems have been reported in the literature [1][3]. Peng and Shin proposed a new scheduling algorithm for tasks with precedence constraints in distributed systems [9]. Arpaci-Dusseau introduced two key mechanisms in implicit coscheduling for distributed systems [1]. The above algorithms were designed for homogeneous distributed systems.

In recent years, the issue of scheduling on heterogeneous distributed systems has been addressed and reported in the literature [5][16]. Ranaweera and Agrawal developed a scalable scheduling scheme called STDP for heterogeneous systems [11]. Srinivasan and Jha incorporated reliability cost, defined to be the product of processor failure rate and task execution time, into scheduling algorithms for tasks with precedence constraints [15]. A. Dogan and F. Özgüner studied reliable matching and scheduling for tasks with precedence constraints in heterogeneous distributed systems. Due to the lack of security awareness, these algorithms are not suitable for security-sensitive distributed computing applications. On the other hand, however, an increasing number of mission-critical applications with high demands of quality of security service have been emerging in various distributed systems [4][7][13][14]. For example, in a real-time stock quote update and trading web service system, each incoming request from business partners and each outgoing response from an enterprise's back-end application have deadlines and security requirements, which have to be dealt with by a server located between the business partners and enterprise backend applications [17]. Furthermore, the server can judiciously select a suitable security level from the range of security service levels, which are predefined combinations of transport and message security mechanisms. Typical security levels in a real-time quote and trading system are [17]: Routing + message security; Routing + SSL; Routing + SSL + message security; Routing + SSL + client authentication and Routing + SSL + message security + client authentication.

In our previous work, we proposed a family of dynamic security-aware scheduling algorithms for a cluster [18] and a Grid [19]. Unfortunately, these scheduling algorithms only support homogeneous computing applications, thus limiting their applicability to heterogeneous distributed systems. Hence, we are motivated in this study to formalize the security heterogeneity concept, and to propose a scheduling algorithm to improve security of heterogeneous distributed systems while minimizing computational overhead.

2 Modeling Tasks and Their Security Requirements

In this study, we consider a queuing architecture of an n-site distributed system in which n heterogeneous sites are connected via a network to process independent tasks submitted by m users. Let $M = \{M_1, M_2, \ldots, M_n\}$ denote the set of heterogeneous sites.

2.1 System Model

The system model, depicted in Figure 1, is composed of a task schedule queue, STAS task scheduler, and *n* local task queues. The function of STAS is intended to make a good task allocation decision for each arrival task to satisfy its security requirements and maintain an ideal performance in conventional performance metrics such as average response time.

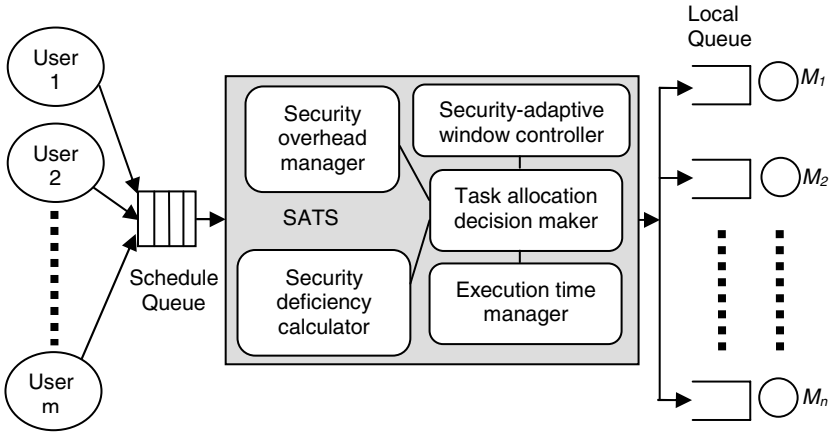


Fig. 1. System model of the SATS strategy

A schedule queue is used to accommodate incoming tasks. SATS scheduler then processes all arrival tasks in a First-Come First-Served (FCFS) manner. After being handled by SATS, the tasks are dispatched to one of the designated site $M_i \in M$ for execution. The sites, each of which maintains a local queue, can execute tasks in parallel. The main component of the system model above is SATS, which is composed of five modules: (1) Execution time manager; (2) Security overhead manager; (3) Degree of security deficiency (*DSD*) calculator; (4) Security-adaptive window controller; and (5) Task allocation decision maker. Since execution time of each task can be estimated by code profiling and statistical prediction [2], we assume that the execution time of each arrival task for each site is a prior and this information is managed in the execution time manager module. Similarly, we assume that the security overhead for each arrival task on each site is a prior, and this information is maintained in the security overhead manager module. The *DSD* calculator is used to calculate discrepancies between an arrival task's security level requirements and the security levels that each site offers. The function of security-adaptive window controller is to vary size of the window to discover a suitable site for the current arrived task so that (1) its security demands can be well met; (2) the total execution time can be as small as possible.

After retrieving information like execution time on each site, security overhead on each site, degree of security deficiency on each site and the size of security-adaptive window for the current task from the corresponding modules, the task allocation decision maker will decide which site will be assigned to the task.

Each site in the system model above is inherently heterogeneous in both computation and security. Computational heterogeneity means that for each task the execution time on different sites is distinctive. While each task has an array of security service requests, each site offers the security services with different levels. The level of a security service provided by a site is normalized in the range from 0.1 to 1.0. Suppose site M_j, offers q security services, $P_j = (p_j^1 , p_j^2 , \ldots, p_j^q)$, a vector of security levels, characterizes the security levels provided by the site. p_j^k is the security level of the kth security service provided by M_j. To meet security requirement, security overhead of the task will be considered. Security heterogeneity suggests that the security overhead of a task varies on each site.

2.2 Modeling Tasks with Security Requirements

We consider a class of heterogeneous distributed systems where an application is comprised of a collection of tasks performed to accomplish an overall mission. It is assumed that tasks are independent of one another. Each task requires a set of security services with various security levels specified by a user. Values of security levels are normalized in the range from 0.1 to 1.0 as well. For example, a task specifies in its request security level 0.7 for the authentication service, 0.3 for the integrity service, and 0.8 for the encryption service. Note that the same security level value in different security services may have various meanings.

Suppose there is a task T_i submitted by a user, T_i is modeled as a set of rational parameters, e.g., $T_i = (a_i, E_i, f_i, l_i, S_i)$, where a_i and f_i are the arrival and finish times, and l_i denotes the amount of data (measured in MB) to be protected. E_i is a vector of execution times for task T_i on each site in M, and $E_i = (e_i^1 , e_i^2 , \ldots, e_i^n)$. Suppose T_i requires q security services, $S_i = (s_i^1 , s_i^2 , \ldots, s_i^q)$, a vector of security levels, characterizes the security requirements of the task. s_i^k is the security level of the kth security service required by T_i. Please note that a way of quantitatively measuring security is still an open question to be answered. Still, we believe that the proposed security requirement model is of help in this research because of the following two reasons. First, although the measurements of security requirements and security levels are not completely objective, performance improvements of the SATS algorithm compared with the two existing approaches in terms of security (i.e., risk-free probability and degree of security deficiency) are still valid because the performance of the three algorithms is evaluated using an identical set of criteria under the same circumstance. Second, quantitatively modelling security requirements and security levels makes it possible for us to compare the security performance of different algorithms and perceive the performance discrepancies among them.

A security-aware scheduler has to make use of a function to measure the security benefits gained by each arrival task. In particular, the security benefit of task T_i is quantitatively modeled as a function of the discrepancy between security levels requested and the security levels offered. The security benefit function for task T_i on site M_j is denoted by $DSD: (S_i, P_j) \rightarrow \Re$, where $\Re$ is the set of non-negative real numbers:

$$DSD(s_i) = \sum_{k=1}^{q} w_i^k \, g(s_i^k, p_j^k), \quad g(s_i^k, p_j^k) = \begin{cases} 0, \text{if } s_i^k \le p_j^k \\ s_i^k - p_j^k, \text{otherwise} \end{cases}, \tag{1}$$

where $0 \le w_i^k \le 1$, $\sum_{k=1}^{q} w_i^k = 1$.

Note that w_i^k is the weight of the kth security service for task T_i. Users specify in their requests the weights to reflect relative priorities given to the required security services. *Degree of Security Deficiency*, or *DSD,* is defined to be a weighted sum of q discrepancy values between security levels requested by a task and the security levels offered by a site. For each task, a small *DSD* value means a high satisfaction degree. Zero *DSD* value implies that a task's security requirements can be perfectly met. That is, there exists at least one site M_j in M that can satisfy the following condition:

$$\forall k \in [1, q], s_i^k \le p_j^k.$$

Let X_i be all possible schedule for task T_i, $x_i \in X_i$ be a scheduling decision of T_i. Given a task T_i, the *degree of security deficiency value* (*DSDV*) of T_i is expected to be minimized:

$$DSDV(x_i) = \min_{x_i \in X_i} \{DSD(x_i)\} = \min_{x_i \in X_i} \left\{ \sum_{k=1}^{q} w_i^k (g(s_i^k, p^k(x_i))) \right\} \tag{2}$$

where $p^k(x_i) = p_j^k$ if the task is allocated to site j. A security-aware scheduler strives to minimize the system's overall *DSDV* value defined as the sum of the degree of security deficiency of submitted tasks (See Equation 1). Thus, the following *DSDV* function needs to be minimized:

$$SDSD(x) = \min_{x \in X} \left\{ \sum_{T_i \in T} DSDV(x_i) \right\}. \tag{3}$$

where T is a set of submitted tasks. Substituting Equation 2 into 3 yields the following security objective function. Thus, our proposed SATS scheduling algorithm makes an effort to schedule tasks in a way to minimize Equation 4:

$$SDSD(x) = \min_{x \in X} \left\{ \sum_{T_i \in T} \left(\min_{x_i \in X_i} \left\{ \sum_{k=1}^{q} w_i^k (f(s_i^k, p^k(x_i))) \right\} \right) \right\}. \tag{4}$$

Since the degree of security deficiency for task T_i merely reflects the security service satisfaction degree experienced by the task, it is inadequate to measure quality of security for T_i during its execution. Therefore, we derive in this section the probability $P_{rf}(T_i, M_j)$ that T_i remains risk-free during the course of its execution.

The quality of security of a task T_i with respect to the kth security service is calculated as $\exp\left(-\lambda_i^k \left(e_i^j + \sum_{l=1}^{q} c_{ij}^l (s_i^l)\right)\right)$ where λ_i^k is the task's risk rate of the

*k*th security service, and $c_{ij}^{l}(s_i^l)$ is the security overhead experienced by the task on site *j*. The risk rate is expressed as:

$$\lambda_i^k = 1 - \exp\left(-\alpha(1 - s_i^k)\right). \tag{5}$$

Note that this model assumes that risk rate is a function of security levels, and the distribution of risk-free 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 task T_i on site M_j can be obtained below by considering all security services provided to the task. Consequently, we have:

$$P_{rf}(T_i, M_j) = \prod_{k=1}^{q} \exp(-\lambda_i^k\left(e_i^j + \sum_{l=1}^{q} c_{ij}^l(s_i^l)\right)) = \exp\left(-\left(e_i^j + \sum_{l=1}^{q} c_{ij}^l(s_i^l)\right)\sum_{k=1}^{q}\lambda_i^k\right). \tag{6}$$

Using equation 6, we obtain the overall quality of security of task T_i in the system as follows,

$$P_{rf}(T_i) = \sum_{j=1}^{n}\{P[x_i = j]\cdot P_{rf}(T_i, M_j)\} = \sum_{j=1}^{n}\left\{p_{ij}\cdot\exp\left(-\left(e_i^j + \sum_{l=1}^{q} c_{ij}^l(s_i^l)\right)\sum_{k=1}^{q}\lambda_i^k\right)\right\} \tag{7}$$

where p_{ij} is the probability that T_i is allocated to site M_j. Given a task set T, the probability that all tasks are free from being attacked during their executions is computed based on Equation 7. Thus,

$$P_{rf}(T) = \prod_{T_i \in T} P_{rf}(T_i). \tag{8}$$

By substituting the risk rate model into Equation 8, we finally obtain $P_{rf}(X)$ as shown below:

$$P_{rf}(X) = \prod_{T_i \in T}\left\{\sum_{j=1}^{n}\left\{p_{ij}\cdot\exp\left(-\left(e_i^j + \sum_{l=1}^{q} c_{ij}^l(s_i^l)\right)\sum_{k=1}^{q}\lambda_i^k\right)\right\}\right\}. \tag{9}$$

In summary, DSD values show us security service satisfaction degrees experienced by tasks, while risk-free probability measured by Equation 9 defines quality of security provided by a heterogeneous distributed system. In Section 5 these two metrics are used to evaluate security of distributed systems.

2.3 Heterogeneity Model

The computational weight of task T_i on site M_j (e.g., w_i^j) is defined as a ratio between its execution time on M_j and that on the fastest site in the system. That is, we have $w_i^j = e_i^j \Big/ \min_{k=1}^{n}\left(e_i^k\right)$. The computational heterogeneity level of task T_i, referred to as

H_i^C, can be quantitatively measured by the standard deviation of the computational weights. Formally, H_i^C is expressed as:

$$H_i^C = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(w_i^{avg} - w_i^j\right)^2}\ , \text{where } w_i^{avg} = \left(\sum_{j=1}^{n} w_i^j\right) \Big/ n \cdot \tag{10}$$

The computational heterogeneity of a task set T can be computed by $H^C = \frac{1}{|T|}\sum_{T_i \in T} H_i^C$.

There are three types of security heterogeneities. (i) Security heterogeneity of a particular task T_i indicates the difference of security requirements in the q security services requested by the task (see Equation 11). (ii) Security heterogeneity of a given security service provided by each site in a distributed system reflects the discrepancy of the offered security levels of the service in the system (see Equation 12). (iii) Security heterogeneity of a particular site M_j shows the deviation of the q security levels provided by the site (see Equation 13).

Given a task T_i and its security requirement $S_i = (s_i^1, s_i^2, \ldots, s_i^q)$, the heterogeneity of security requirement for T_i is measured by the standard deviation of the security levels in the vector. Thus,

$$H_i^S = \sqrt{\frac{1}{q}\sum_{j=1}^{q}\left(s_i^{avg} - s_i^j\right)^2}\ , \text{where } s_i^{avg} = \left(\sum_{j=1}^{q} s_i^j\right) \Big/ n \cdot \tag{11}$$

The security requirement heterogeneity of a task set T can be computed by $H^S = \frac{1}{|T|}\sum_{T_i \in T} H_i^S$.

The heterogeneity level of the kth security service in a distributed system is defined as:

$$H_k^V = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(p_{avg}^k - p_i^k\right)^2}\ , \text{where } p_{avg}^k = \left(\sum_{i=1}^{n} p_i^k\right) \Big/ n \cdot \tag{12}$$

Using Equation 12, the heterogeneity of security services can be written as $H^V = \frac{1}{q}\sum_{k=1}^{q} H_k^V$.

Finally, the heterogeneity level of security services in site M_j is defined to be:

$$H_j^M = \sqrt{\frac{1}{q}\sum_{k=1}^{q}\left(p_j^{avg} - p_j^k\right)^2}\ , \text{where } p_j^{avg} = \left(\sum_{k=1}^{q} p_j^k\right) \Big/ q \cdot \tag{13}$$

2.4 Security Overhead Model

Now we consider security overhead incurred by security services. The following security overhead model includes three services, namely, encryption, integrity, and authentication [18]. The security overhead model can be easily extended to incorporated more security services.

Suppose task T_i requires q security services, which are provided in sequential order. Let s_i^k and $c_{ij}^k(s_i^k)$ be the security level and overhead of the kth security service, the security overhead c_{ij} experienced by T_i on site M_j can be computed using Equation 14. In particular, the security overhead of T_i with security requirements for the three services above is modelled by Equation 15.

$$c_{ij} = \sum_{k=1}^{q} c_{ij}^k(s_i^k), \text{ where } s_i^j \in S_i. \tag{14}$$

$$c_{ij} = \sum_{k\in\{a,e,g\}} c_{ij}^k(s_i^k), \text{ where } s_i^j \in S_i. \tag{15}$$

where $c_{ij}^e(s_i^e)$, $c_{ij}^g(s_i^g)$, and $c_{ij}^a(s_i^a)$ are overheads caused by the authentication, encryption, and integrity services [18]. Our security level assignment is reasonable because a security mechanism providing higher quality of security imposes higher overhead than mechanisms offering lower security.

The encryption overhead c_i^e of T_i on M_j is computed using Equation 16, where π_i^e is the CPU time spent in encrypting security sensitive data.

$$c_{ij}^e(s_i^e) = \pi_{ij}^e s_i^e, \text{ where } s_i^e \in S_i. \tag{16}$$

The integrity overhead can be calculated using the following equation, where l_i is the amount of security sensitive data, and $\mu^g(s_i^g)$ is a function mapping a security level into its corresponding integrity service performance.

$$c_{ij}^g(s_i^g) = l_i / \mu^g(s_i^g), \text{ where } s_i^g \in S_i. \tag{17}$$

The security level of a security mechanism can be quantitatively measured by the amount of cost needed to successfully break the mechanism. However, quantitatively measuring the security level of a security mechanism is a nontrivial research issue, and it is out of the scope of this work.

3 The SATS Algorithm

For task T_i, the earliest start time on site M_j is $es_j(T_i)$, which can be computed by Equation 18:

$$\text{es}_j(T_i) = r_j + \sum_{T_l \in W_j} \left(e_l^j + \sum_{k=1}^{q} c_{lj}^k(s_l^k) \right) \tag{18}$$

where r_j represents the remaining overall execution time of a task currently running on the jth site, and $e_l^j + \sum_{k=1}^{q} c_{lj}^k(s_l^k)$ is the overall execution time (security overhead is factored in) of waiting task T_l assigned to site M_j prior to the arrival of T_i. Thus, the earliest start time of T_l is a sum of the remaining overall execution time of the running

task and the overall execution times of the tasks with earlier arrival on site M_j. Therefore, the earliest completion time for task T_i on site M_j can be calculated as:

$$ec_j(T_i) = es_j(T_i) + e_i^j + \sum_{k=1}^{q} c_{ij}^k(s_i^k) = r_j + \sum_{T_l \in W_j} \left(e_l^j + \sum_{k=1}^{q} c_{lj}^k(s_l^k) \right) + e_i^j + \sum_{k=1}^{q} c_{ij}^k(s_i^k) \quad (19)$$

```
1.for each task T_i submitted to the schedule queue do
2.    for each site M_j in the system do
3.        Use Eq.18 to compute es_j(T_i), the earliest start time of T_i on site M_j;
4.        Use Eq.19 to compute ec_j(T_i), the earliest completion time of T_i on M_j;
5.    end for
6.    Sort all sites in earliest completion time in a non-decrease order
7.    for each site in the security-adaptive window do
8.        Use Eq.1 to compute DSD(s_i) /* Compute DSD for each site*/
9.    end for
10.   Select the site M_r that can offer the smallest DSD value and assign T_i on it
11.   Update site M_r's earliest available time es_j
12.   Use Eq.6 to compute risk-free probability for task T_i
13.   Record start time and completion time for task T_i
14.end for
```

Fig. 2. The SATS algorithm

The SATS algorithm is outlined in Figure 2. The goal of the algorithm is to deliver optimal quality of security while maintaining high performance for tasks running on heterogeneous systems. To achieve the goal, SATS manages to minimize degree of security deficiency (see Equation 1) of each task (see Step 10 in Fig. 2) without performance deterioration. Before optimizing the degree of security deficiency of task T_i, SATS sorts all the sites in a non-decrease order in T_i's total execution time based on the information retrieved from the execution time manager and the security overhead manager (see Figure 1). Step 7 computes the degree of security deficiencies for the task on each site in the light of the security deficiency calculator. Combining the input from the security-adaptive window controller, the task allocation decision maker decides a site to which the task is allocated.

4 Simulations

Using extensive simulation experiments based on San Diego Supercomputer Center (SDSC) SP2 log, we evaluate in this section the potential benefits of the SATS algorithm. The real trace was sampled on a 128-node (66MHz) IBM SP2 from May 1998 through April 2000. To simplify our experiments, we utilized the first three months data with 6400 parallel tasks in simulation. Since the trace was sampled from a homogenous environment, to reflect the heterogeneity of the simulated distributed system, we translated the "execution time" of each task from a single value to a vector with n (number of sites) elements based on the heterogeneity model described in Section 2.3. In purpose of revealing the strength of SATS, we compared it with two well-known scheduling algorithms, namely, *Min-Min* and *Sufferage* [14]. *Min-Min* and *Sufferage* are non-preemptive task scheduling algorithms, which schedule a

stream of independent tasks onto a heterogeneous distributed computing system. The two algorithms are briefly described below.

(1) MINMIN: For each submitted task, the site that offers the earliest completion time is tagged. Among all the mapped tasks, the one that has the minimal earliest completion time is chosen and then allocate to the tagged site.
(2) SUFFERAGE: Allocating a site to a submitted task that would "suffer" most in terms of completion time if that site is not allocated to it.

Table 1. Characteristics of system parameters

Parameter	Value (Fixed) - (Varied)
Number of tasks	(6400)
Number of sites	(16) – (8, 16, 32,64,128)
Task arrival rate	Decided by the trace
Size of security-adaptive window	(8) – (1, 2, 4, 8, 16)
Site security level (uniform dist.)	(0.1 – 1.0)
Task security level (uniform dist.)	(0.1 – 1.0)
Weights of security services	Authentication=0.2, Encryption=0.5, Integrity =0.3

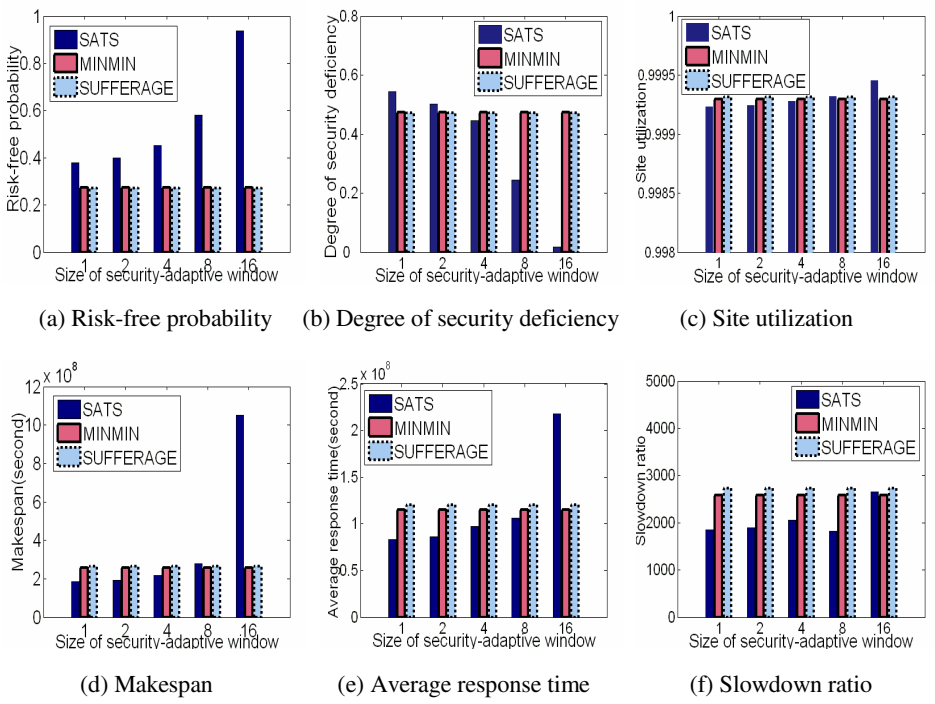
4.1 Simulator and Simulation Parameters

The parameters of sites in the simulated distributed system are chosen to resemble real-world workstations like IBM SP2 nodes. Table 1 summarizes the key configuration parameters of the simulated distributed system used in our experiments.

We modified the trace by adding a block of data to be secured for each task in the trace. The size of the security-required data assigned to each task is controlled by a uniform distribution (see Table 1). Although "task number", "submit time", and "execution time" of tasks submitted to the system are taken directly from the trace, "size of data to be secured", "number of sites", "computational heterogeneity", and "security heterogeneity" are synthetically generated in accordance with the above model since they are not available in the trace. The performance metrics we used include: *risk-free probability* (see Equation 9), *degree of security deficiency* (see Equation 1), *site utilization* (defined as the percentage of total task running time out of total available time of a given site), *makespan* (the latest task completion time in the task set), *average response time* (the response time of a task is the time period between the task's arrival and its completion and the average response time is the average value of all tasks' response time), *slowdown ratio* (the slowdown of a task is the ratio of the task's response time to its service time and the slowdown ratio is the average value of all tasks' slowdowns).

4.2 Overall Performance Comparisons

The goal of this experiment is two fold: (1) to compare the proposed SATS algorithm against the two heuristics, and (2) to understand the sensitivity of SATS to the size of security-adaptive window.



(a) Risk-free probability (b) Degree of security deficiency (c) Site utilization

(d) Makespan (e) Average response time (f) Slowdown ratio

Fig. 3. Performance impact of size of security-adaptive window

Figure 3 shows the simulation results for the three algorithms on a distributed system with 16 sites. Since MINMIN and SUFFERAGE do not have a security-adaptive window, their performance in all six metrics keeps constant. We observe from Figure 3a that SATS significantly outperforms the two heuristics in terms of risk-free probability, whereas MINMIN and SUFFERAGE algorithms exhibit similar performance.

5 Summary and Future Work

In this paper, we considered the security requirements of applications in the context of task scheduling in heterogeneous distributed systems because an increasing number of applications running on heterogeneous distributed systems requires not only descent scheduling performance but also high quality of security. To solve this problem, we proposed a security-adaptive scheduling heuristic that is based on the concept of security heterogeneity. Experimental results demonstrate that our strategy outperforms existing approaches in both security and performance. In future research, the heuristic will be extended to schedule parallel applications.

References

1. Arpaci-Dusseau, A.C.: Implicit Coscheduling: Coordinated Scheduling with Implicit Information in Distributed Systems. ACM Trans. on Computer Systems, Vol.19, No.3, (2001) 283-331
2. Braun , T. D. et al.: A Comparison Study of Static Mapping Heuristics for a Class of Meta-tasks on Heterogeneous Computing Systems. Proc. Workshop Heterogeneous Computing, (1999) 15-29
3. Casavant, T.L, Kuhl, J.G.: A Taxonomy of Scheduling in General-purpose Distributed Computing Systems. IEEE Trans. Software Engineering, Vol.14, No.2, (1988) 141-154
4. Connelly, K., Chien, A.A.: Breaking the barriers: high performance security for high performance computing. Proc. Work-shop on New security paradigms, Sept. (2002)
5. Dogan, A., Özgüner, F.: Reliable matching and scheduling of precedence-constrained tasks in heterogeneous distributed computing. Proc. Int'l Conf. Parallel Processing, (2000) 307-314
6. Dogan, A., Özgüner, F.: LDBS: A Duplication Based Scheduling Algorithm for Heterogeneous Computing Systems. Proc. Int'l Conf. Parallel Processing, (2002) 352-359
7. Donoho, G.: Building a Web Service to Provide Real-Time Stock Quotes. MCAD.Net, Feb. (2004)
8. Lampson, B., Abadi, M., Burrows, M., Wobber, E.: Authentication in distributed systems: Theory and practice. ACM Trans. Computer Systems, Vol.10, No.4, Nov. (1992) 265-310
9. Peng, D.-T., Shin, K.G.: Optimal scheduling of cooperative tasks in a distributed system using an enumerative method. IEEE Trans. Software Engineering, Vol.19, No.3, Mar. (1993) 253-267
10. Petrini F., Feng, W.-C.: Scheduling with Global Information in Distributed Systems. Proc. 20th Int'l Conf. Distributed Computing Systems, April (2000) 225 – 232
11. Ranaweera, S., Agrawal, D.P.: Scheduling of Periodic Time Critical Applications for Pipelined Execution on Heterogeneous Systems. Proc. Int'l Conf. Parallel Processing, Sept. (2001) 131-138
12. Sandhu, R.S. et al.: Role-Based Access Control Models. IEEE Computer, Vol.29,No.2, (1996) 38-47
13. Son, S.H., Mukkamala, R., David, R.: Integrating security and real-time requirements using covert channel capacity. IEEE Trans. Knowledge and Data Engineering. Vol.12, No.6, (2000) 865 – 879
14. Song, S., Kwok, Y.-K., Hwang, K.: Trusted Job Scheduling in Open Computational Grids: Security-Driven Heuristics and A Fast Genetic Algorithms. Proc. Int'l Symp. Parallel and Distributed Processing, (2005)
15. Srinivasn, S., Jha, N.K.: Safety and Reliability Driven Task Allocation in Distributed Systems. IEEE Trans. Parallel and Distributed Systems, Vol.10, No.3, Mar. (1999) 238-251
16. Topcuoglu, H., Hariri, S., Wu, M.-Y.: Performance-effective and Low-complexity Task Scheduling for Heterogeneous Computing. IEEE Trans. Parallel and Distributed Sys., Vol.13, No.3, Mar. (2002)
17. VeriSign Corp.: Simplifying Application and Web Services Security - VeriSign Trust Gateway. (2003)
18. Xie, T., Qin, X.: Scheduling Security-Critical Real-Time Applications on Clusters. IEEE Transactions on Computers, vol. 55, no. 7, July (2006) 864-879
19. Xie, T., Qin, X.: Enhancing Security of Real-Time Applications on Grids through Dynamic Scheduling. Proc. 11th Workshop Job Scheduling Strategies for Parallel Processing, MA, June (2005)