Cluster Comput (2007) 10: 145–153
DOI 10.1007/s10586-007-0015-x

Security-driven scheduling for data-intensive applications on grids

Tao Xie · Xiao Qin

Published online: 15 March 2007

Abstract Security-sensitive applications that access and generate large data sets are emerging in various areas including bioinformatics and high energy physics. Data grids provide such data-intensive applications with a large virtual storage framework with unlimited power. However, conventional scheduling algorithms for data grids are unable to meet the security needs of data-intensive applications. In this paper we address the problem of scheduling data-intensive jobs on data grids subject to security constraints. Using a security- and data-aware technique, a dynamic scheduling strategy is proposed to improve quality of security for data-intensive applications running on data grids. To incorporate security into job scheduling, we introduce a new performance metric, degree of security deficiency, to quantitatively measure quality of security provided by a data grid. Results based on a real-world trace confirm that the proposed scheduling strategy significantly improves security and performance over four existing scheduling algorithms by up to 810% and 1478%, respectively.

Keywords Scheduling · Security-sensitive application · Data grid · Degree of security deficiency

T. Xie (✉)
Department of Computer Science, San Diego State University, San Diego, CA 92182, USA
e-mail: xie@cs.sdsu.edu

X. Qin
Department of Computer Science, New Mexico Institute of Mining and Technology, Socorro, NM 87801, USA
e-mail: xqin@cs.nmt.edu

1 Introduction

A grid is a collection of geographically dispersed computing resources, providing a large virtual computing system to users [2]. There are four commonly used kinds of resources in grids: computation, storage, software, and communications. Data grids have been developed to integrate heterogeneous data archives stored in a large number of geographically distributed sites into a single virtual data management system [1]. Furthermore, data grids provide diverse services to fit the needs of high-performance and data-intensive computing [6].

There have been some efforts to develop data-intensive applications like bioinformatics and high energy physics in data grid environments [7, 10]. In recent years, many studies addressed the issue of scheduling for data grids [5, 7]. Although existing scheduling algorithms provide high performance for data-intensive applications, these algorithms are not adequate for data-intensive applications that are security-sensitive in nature.

Security services are critically important to a variety of security-sensitive applications on grids in general [9] and data grids in particular. This is because sensitive data and processing require special safeguard and protection against unauthorized access. Much attention has been focused on security for applications running on grids [3, 4, 9]. Our proposed scheduling strategy for security-sensitive jobs is complementary to existing security techniques for grids in the sense that an additional improvement in security can be achieved by combining our strategy with the existing grid security services.

Very recently, Song et al. proposed security-driven scheduling algorithms for grids [8]. We proposed an array of security-aware scheduling algorithms for applications with timing constraints on single machines 0, clusters [11], and

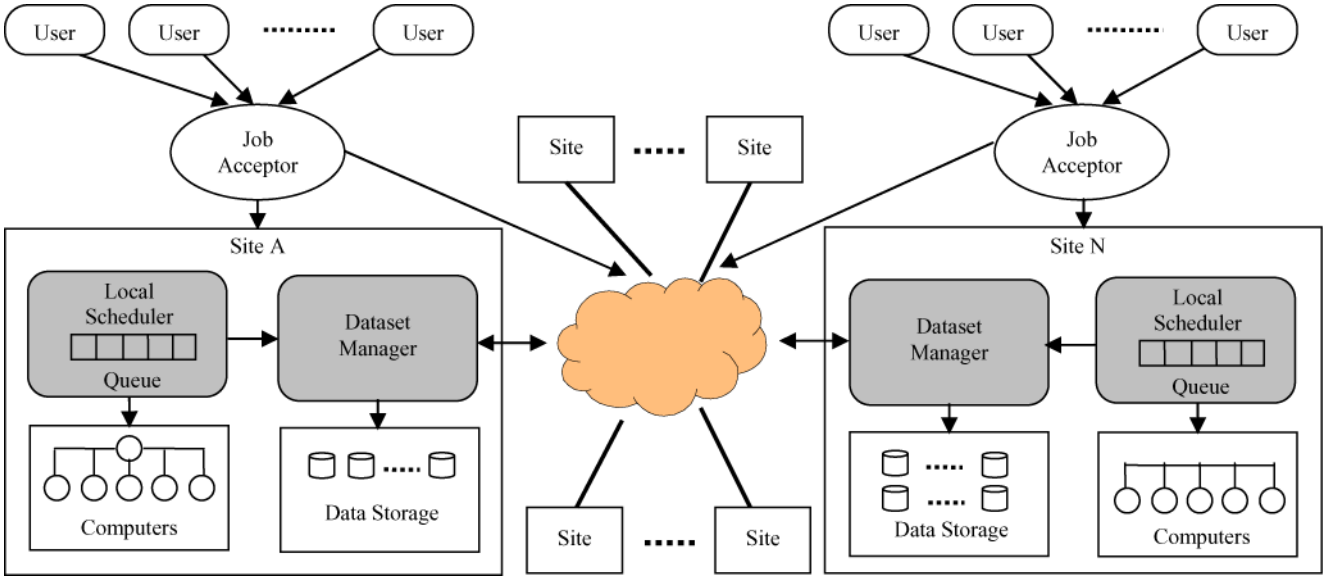


Fig. 1 System model of a data grid

grids [12]. The above algorithms can improve security and performance of various applications. However, these algorithms have no inherent capability of supporting data girds due to the ignorance of heterogeneous resources and data sets read/write by applications.

In data grids, a growing number of applications have both security and data constraints. In this regard, we are motivated to introduce a new concept of degree of security deficiency. Additionally, a scheduling strategy is proposed to enhance security of data-intensive applications on data grids. The main contributions of this study include: (1) a model for data-intensive applications with security requirements; (2) a concept of degree of security deficiency; (3) considerations of datasets accessed and generated by applications; (4) integration of security heterogeneity into job scheduling; (5) a scheduling strategy for data-intensive applications on data grids.

The rest of the paper is organized in the following way. Section 2 presents a system model used to construct the scheduling strategy. The risk-free probability and the scheduling strategy are detailed in Sect. 3. Section 4 discusses experimental results that confirm the validity of the proposed scheduling strategy. Section 5 concludes the paper with summary and future research directions.

2 System and job models

We model a data grid as a network of n heterogeneous sites with various system architectures. Each site in the data grid is comprised of data stores and computational facilities. Let $M = \{M_1, M_2, \ldots, M_n\}$ denote a set of sites, each of which represents a pair $M_i = (N_i, D_i, P_i)$, where N_i is a set of nodes residing in the site, D_i is an array of data sets accessed and generated by data-intensive jobs, $P_j = (p_j^1, p_j^2, \ldots, p_j^q)$ is a vector characterizing the security levels provided by the site.

Figure 1 shows the system model of a data grid. Each site has a job acceptor that accommodates arrival jobs submitted by multiple users. If a job is allocated to a remote site, it will be dispatched to the other site through a high-speed network. Otherwise, it will be placed into a local queue managed by a local scheduler. The function of dataset manager is three-fold. (1) It monitors local datasets. If the popularity of a dataset exceeds a threshold value, the dataset manager will automatically replicate the dataset to a remote site. (2) It accepts dataset requests issued by the local scheduler. When a job needs a remote dataset to run, the local scheduler passes a dataset request to the dataset manager. (3) If a dataset request is sent from the local scheduler, the dataset manager will deliver the request to a destination site.

Each data-intensive jobs considered in this study consists of a collection of tasks performed to accomplish an overall mission. Without loss of generality, we assume that tasks in a job are independent of one another. A job is modeled as a set of parameters, e.g., $J = (a, u, E, c, d, L, SC, SD)$, where a is the arrival times, u is the number of requested nodes, c is the size of the job code, d denotes the size of input/output data set, and L represents a group of sites where the data set is stored. E is a vector of execution times for the job on each site in M, and $E_i = (e_i^1, e_i^2, \ldots, e_i^n)$. The security requirement of job J is modeled by a vector of security levels, e.g., $SC = (sc^1, sc^2, \ldots, sc^q)$, where q is the number of required security services. sc^k is the required security level of the kth security service. Similarly, the security re-

quirement of the job's data set is modeled by a vector of security levels, e.g., $SD = (sd^1, sd^2, \ldots, sd^q)$.

3 Security overhead model

We introduce in this section a security overhead model for data-intensive jobs running on a data grid. Detailed information about the security overhead model can be found in [12].

First, we consider a case where input data is locally available and processing is performed on the local site. Let sc_i^k and $c_j^k(sc_i^k)$ be a security level and the overhead of the kth security service for job J_i. Likewise, let sd_i^k and $c_j^k(sd_i^k)$ denote a security level and the overhead of the kth security service for the job's data set. The security overhead c_{ij} experienced by job J_i on local site M_i where the data set is available can be computed as a sum of times spent in securing the application code and data set. Thus, we have the following equation, where $sc^k \in SC$ and $sd^k \in SD$.

$$c_{ij} = \sum_{k=1}^{q} c_j^k\left(sc_i^k\right) + \sum_{k=1}^{q} c_j^k\left(sd_i^k\right) = \sum_{k=1}^{q}\left[c_j^k\left(sc_i^k\right) + c_j^k\left(sd_i^k\right)\right]. \quad (1)$$

Second, we derive an expression to calculate the security overhead of a locally executed job J_i accessing its remote data set. In this scenario, job's data set needs to be fetched from the remote site through networks. Suppose there are p number of security services provided for a link between site v and site j. The security overhead of the kth security service for the data set delivered from site v to site j is expressed as $c_{vj}^k(sd_i^k)$. The total security overhead is the sum of the security overhead caused by data transfer, application code, and data set protections. Therefore, the security overhead in this case can be written as:

$$c_{ij} = \sum_{k=1}^{p} c_{vj}^k\left(sd_i^k\right) + \sum_{k=1}^{q}\left[c_j^k\left(sc_i^k\right) + c_j^k\left(sd_i^k\right)\right]. \quad (2)$$

Third, Expression (3) is used to compute the security overhead of a remotely executed job J_i that accesses its data set on a local site. Thus, the application code needs to be transmitted to the remote site where the data is stored. The security overhead of the kth security service for transmitting the application code from remote site v to local site j is denoted by $c_{vj}^k(sc_i^k)$. Finally, the total security overhead in this case can be calculated as:

$$c_{ij} = \sum_{k=1}^{p} c_{vj}^k\left(sc_i^k\right) + \sum_{k=1}^{q}\left[c_j^k\left(sc_i^k\right) + c_j^k\left(sd_i^k\right)\right], \quad (3)$$

where the two terms on the right-hand side of (3) are the security overhead incurred in securing the application code transfer, application code, and data set.

Last, we consider the security overhead of a job executed on a remote site to which the job's data set is moved. In this scenario, the security overhead includes two major categories of cost: (a) overhead of security services for transmitting the application code and data set; (b) time spent in securing the job's code and data set. Thus we have the following equation:

$$c_{ij} = \sum_{k=1}^{p}\left[c_{vj}^k\left(sc_i^k\right) + c_{vj}^k\left(sd_i^k\right)\right] + \sum_{k=1}^{q}\left[c_j^k\left(sc_i^k\right) + c_j^k\left(sd_i^k\right)\right]. \quad (4)$$

4 Scheduling strategy

4.1 Degree of security deficiency

Before we proceed to the design of the scheduling strategy, we need to construct a model to measure quality of security provided by a data grid. Specifically, we introduce a new concept of security deficiency quantified as the discrepancy between requested security levels and offered security levels. Thus, the security deficiency of the kth security service for job J_i on site N_j is:

$$\delta_{site}\left(sc_i^k, sd_i^k, p_j^k\right) = \begin{cases} 0, \text{ if } sc_i^k \le p_j^k \text{ and } sd_i^k \le p_j^k \\ sc_i^k - p_j^k, \text{ if } sc_i^k > p_j^k \text{ and } sd_i^k \le p_j^k \\ sd_i^k - p_j^k, \text{ if } sc_i^k \le p_j^k \text{ and } sd_i^k > p_j^k \\ sc_i^k + sd_i^k - 2p_j^k, \text{ otherwise} \end{cases}. \quad (5)$$

With regard to the kth security service, the security deficiency of a data set delivered from site v to site j is defined by the following equation.

$$\delta_{link}\left(sc_i^k, p_{vj}^k\right) = \begin{cases} 0, \text{ if } sc_i^k \le p_{vj}^k \\ sc_i^k - p_{vj}^k, \text{ otherwise} \end{cases}, \quad (6)$$

where p_{vj}^k is the security levels provided by the link.

Similarly, the security deficiency for application code transmitted from site v to site j is expressed as:

$$\delta_{link}\left(sd_i^k, p_{vj}^k\right) = \begin{cases} 0, \text{ if } sd_i^k \le p_{vj}^k \\ sd_i^k - p_{vj}^k, \text{ otherwise} \end{cases}. \quad (7)$$

A small security deficiency indicates a high degree of service satisfaction. The security deficiency of a job J_i on site M_j can be derived from the following equation.

$$SD_{ij} = \sum_{k=1}^{q}\left[w_i^k \delta_{site}\left(sc_i^k, sc_i^k, p_j^k\right)\right]$$

$$+\sum_{v=1}^{n}\left[xc_{vj}\sum_{k=1}^{p}\left[w_i^k\delta_{link}\left(sc_i^k, p_{vj}^k\right)\right]\right]$$
$$+\sum_{v=1}^{n}\left[xd_{vj}\sum_{k=1}^{p}\left[w_i^k\delta_{link}\left(sd_i^k, p_{vj}^k\right)\right]\right], \quad (8)$$

where xc_{vj} and xd_{vj} are two step functions. $xc_{vj} = 1$ if the application code is moved from site v to j, otherwise $xc_{vj} = 0$. $xd_{vj} = 1$ if the dataset is fetched from site v to j, otherwise $xd_{vj} = 0$. w_i^k is the weight of the kth security service, e.g., $0 \le w_i^k \le 1$, and $\sum_{k=1}^{q} w_i^k = 1$. Note that weights reflect relative priorities given to security services.

Given a sequent of data-intensive jobs J, the *degree of security deficiency* for a data grid M under allocation X is defined as the sum of the security deficiencies of all the jobs. The degree of security deficiency can be written as:

$$DSD(M, J, X) = \sum_{\forall J_i}\sum_{j=1}^{n}[x_{ij}SD_{ij}], \quad (9)$$

where $x_{ij} \in X$, $x_{ij} = 1$ if J_i is allocated to site M_j, $\sum_{j=1}^{n} x_{ij} = 1$.

4.2 Scheduling strategy description

Given a data grid M and a sequence of jobs J, our scheduling strategy is intended to generate an allocation X minimizing the degree of security deficiency computed by (9). Finally, we can obtain the following non-linear optimization problem formulation:

$$\text{minimize } DSD(M, J, X) = \sum_{\forall J_i}\sum_{j=1}^{n}[x_{ij}SD_{ij}]$$

$$\text{subject to } J_i \in J, \sum_{j=1}^{n} x_{ij} = 1.$$

Now we present the security- and heterogeneity-aware scheduling strategy (SAHA). The earliest start time of job j_i on site M_j can be approximated as below:

$$\text{es}_j(J_i) = r_j + \sum_{J_l \in M_j}\left(e_l^j + \sum_{k=1}^{q}\left[c_{lj}^k\left(sc_l^k\right) + c_{lj}^k\left(sd_l^k\right)\right]\right), \quad (10)$$

where r_j represents the finish time of a job currently running on the jth site, and the second term on the right-hand side is the overall execution time (security overhead is factored in) of waiting jobs assigned to site M_j prior to the arrival of J_i. In other words, the earliest start time of J_i is the sum of the finish time of the running task and the overall execution times of the jobs with earlier arrival on site M_j.

The SAHA strategy is outlined in Fig. 2. The goal is to maximize security while maintaining high performance for data-intensive jobs running on data grids. In order to improve security, SAHA makes an effort to minimize degree of security deficiency measured by (9) (see Steps 9–10 in Fig. 2) without performance deterioration.

1. **for** each job J_i submitted to site M_j **do**
2. **for** each site M_k in the system **do**
3. Use **Equation (9)** to obtain degree of security deficiency;
4. Use **Equation (10)** to compute the earliest start time;
5. Use **Equations (1)-(4)** to compute security overhead;
6. Computer the finish time of J_i on site M_j
7. **end for**
8. Sort all sites in earliest finish time;
9. **for** a group sites providing the minimal finish time **do**
10. Find site M_t that minimize the degree of security deficiency;
11. Update J_i's start and finish times, and schedule information;
12. Dispatch J_i and its dataset to site M_t for execution;
13. **end for**

Fig. 2 The SAHA strategy

Before optimizing the degree of security deficiency, Step 3 manages to calculate the degree of security deficiency for the submitted job on each site. SAHA sorts all the sites in a non-decrease order of earliest finish times estimated by (10) (see Steps 4 and 8). From the group of sites providing the minimal finish times, Step 10 chooses the most appropriate site that substantially reduces the degree of security deficiency. Step 11 is intended to update information pertinent to scheduling decisions, whereas Step 12 dispatches the job and its dataset to the selected site for execution.

4.3 Risk-free probability

We derive in this subsection the probability $P_{rf}(J_i, M_j)$ that J_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 $\exp(-\lambda_i^k(e_i^j + c_{ij}))$ where λ_i^k is the task's risk rate of the kth security service, and c_{ij} is the security overhead experienced by the job on site j. The risk rate is expressed as:

$$\lambda_i^k = 1 - \exp\left(-\alpha\left(1 - s_i^k\right)\right). \quad (11)$$

The quality of security of J_i on site M_j can be obtained below by considering all security services.

$$P_{rf}\left(J_i, M_j\right) = \prod_{k=1}^{q}\exp\left(-\lambda_i^k\left(e_i^j + c_{ij}\right)\right)$$
$$= \exp\left(-\left(e_i^j + c_{ij}\right)\sum_{k=1}^{q}\lambda_i^k\right). \quad (12)$$

Table 1 Characteristics of system parameters

Parameter	Value (fixed)–(varied)
Number of jobs	(6400)–
Number of sites	(32)–(32, 64,128)
Number of datasets	(200)–(100, 200, 400)
Job arrival rate	Decided by the trace
Network bandwidth	1 Gbps (billions of bits/s)
Data to be secured	1–10 MB short, 10–50 MB medium, 50–100 MB long
Size of datasets	(500–800 MB short jobs, 800 MB–1 GB medium jobs, 1–2 GB long jobs)–(500 MB–2 GB)
Dataset popularity distribution	(Regional unif.)–(Uniform, Normal, Geometric)
Dataset popularity threshold	(10)–(2, 4, 6, 8, 10)
Number of sites for each dataset	(1)–(1, 4, 8)
Size of site group	(8)–(1, 2, 4, 8, 16)
Site security level	(0.1–1.0)
Job security level	(0.1–1.0)
Required security services	Encryption, Integrity and Authentication
Weights security services	Authentication weight = 0.2, Encryption weight = 0.5, Integrity weight = 0.3
Heterogeneity	Computation (1.08)–(0, 1.08, 2.27); Security (0.22)–(0, 0.22, 0.56)

Finally, we obtain the overall quality of security of task T_i in the system as follows,

$$P_{rf}(J_i) = \sum_{j=1}^{n} \{p_{ij} \cdot P_{rf}(J_i, M_j)\}, \tag{13}$$

where p_{ij} is the probability that J_i is allocated to site M_j.

5 Experimental results

We use simulation experiments based on San Diego Supercomputer Center (SDSC) SP2 traces to evaluate benefits of the SAHA strategy. To demonstrate the strengths of SAHA, we compared it with two well-known data grid scheduling algorithms, namely, *JobRandom* and *JobDataPresent* [7], among which JobDataPresent is the best algorithm based on results reported in [7]. The two algorithms are briefly described as below. (1) *JobRandom*: For each submitted job, a randomly selected site is allocated to the job. (2) *JobDataPresent*: For each submitted job, a site that has required datasets is assigned to the job. Table 1 summarizes system parameters of the simulated data grid used in our experiments.

5.1 Simulator and simulation parameters

Dataset popularities are randomly generated with a regional uniform distribution. All submitted jobs in the trace fall into three categories based on their execution times. The categories include short jobs, medium jobs, and long jobs. Accordingly, we assign each category a dataset size range (region). Dataset popularities within each range are uniformly distributed.

We modified the trace by adding a block of data to be secured for each job in the trace. The size of the security-required data assigned to each job is controlled by a uniform distribution. The performance metrics we used include: *degree of security deficiency* (see (1)), *risk-free probability* (see (9)), *Average Data Transferred* (defined as the volume of transferred data per job), *average response time* (the response time of a job is the time period between the job's arrival and its completion and the average response time is the average value of all jobs' response time), *slowdown ratio* (the slowdown of a job is the ratio of the job's response time to its service time and the slowdown ratio is the average value of all jobs' slowdowns), *Average idle time percentage* (an idle time percentage of a site is the ratio of the site's idle time to its last job finish time).

5.2 Overall performance comparisons

The goal of this experiment is to compare the proposed SAHA algorithm against the two heuristics in two dataset replication methods, *DataDoNothing* and *DataRandom*.

Figure 3 shows the simulation results for the three scheduling algorithms on a data Grid with 32 sites. We observe from Fig. 3 that SAHA significantly outperforms the two heuristics in terms of degree of security deficiency and risk-free probability, whereas JobRandom and JobDataPresent algorithms exhibit similar performance. We attribute the performance improvement to the fact that SAHA is a security-sensitive scheduler and judiciously assigns a job to

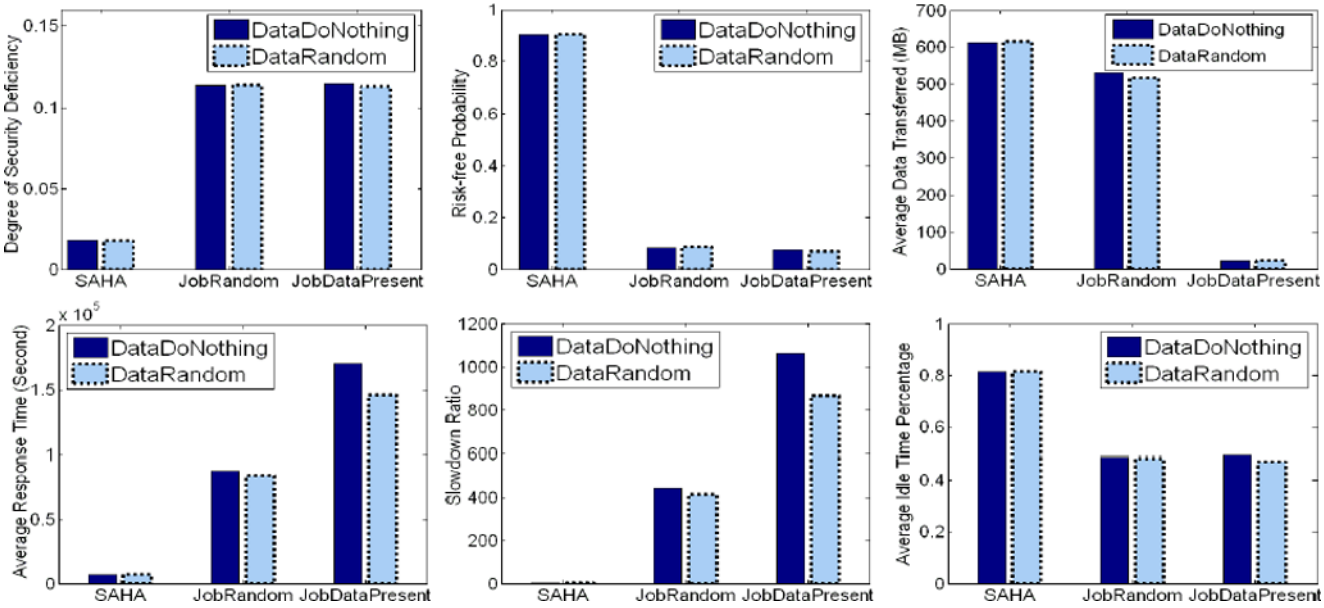


Fig. 3 Overall performance comparisons

a site not only considering its computational time but also its security demands. In addition, SAHA is greatly superior to the two alternatives in conventional performance metrics such as average response time and slowdown ratio, which are mostly concerned by users. The performance improvements of SAHA are at the cost of a higher volume of data transferred. However, the data transmission overhead can be largely alleviated by using a high network bandwidth (see Table 1) that is available in real world. Also, SAHA has a higher average idle time percentage because it can generate an effective schedule that leads to a much shorter average execution time for the job set. The higher idle time percentage suggests that the Grid has potential to accommodate more jobs when the SAHA scheduling algorithm is in place.

5.3 Heterogeneities in computation and security

As we mentioned in Sect. 2, in a heterogeneous distributed system like a data Grid, the computational times of a particular job on different sites are distinctive, which is referred to as computational heterogeneity. Besides, the security overheads incurred by the job on different sites are diverse. We name this security heterogeneity. We believe that both computational and security heterogeneities are essential characteristics of a security-critical data Grid. To differentiate execution time from security overhead, we use the term computational time to refer execution time without security overhead, and the total execution time of a job consists of both computational time and security overhead. In this experiment we investigate the impact of these two heterogeneities on system performance. The three heterogeneity configurations can be found in Table 1. For simplicity, we call SAHA-DataDoNothing SAHADDN and SAHA-DataRandom SAHADR. Similarly, JRDDN stands for JobRandom-DataDoNothing, JRDR stands for JobRandom-DataRandom, JDPDDN stands for JobDataPresent-DataDoNothing, and JDPDR stands for JobDataPresent-DataRandom.

When both computational heterogeneity and security heterogeneity are zero, which means the Grid is homogeneous, SAHADDN and SAHADR noticeably outperforms the other four methods in security and average response time but has the same performance in average idle time percentage (see Fig. 4). Nevertheless, SAHA fully exhibits its power when the two heterogeneities increase as shown in Fig. 4. In the last two heterogeneity configurations, SAHA further performs better than the other two scheduling algorithms JobRandom and JobDataPresent in risk-free probability, average response time, and slowdown ratio. Also, we observe that JobRandom and JobDataPresent noticeably perform worse in average response time, slowdown ratio, and risk-free probability when the heterogeneities increase. This is because they are not heterogeneity-aware, and thus, cannot assign a job to a site that can provide the earliest computational time in most cases.

On the contrary, SAHA can further improve its performance in security and conventional metrics like average response time when the two heterogeneities enlarge. This is because SAHA is a heterogeneity-aware scheduling algorithm, which can factor in computational heterogeneity as well as security overhead heterogeneity when allocates a site for a particular job. Hence, it can always discover a site that can satisfy a job's security needs, while results in a less com-

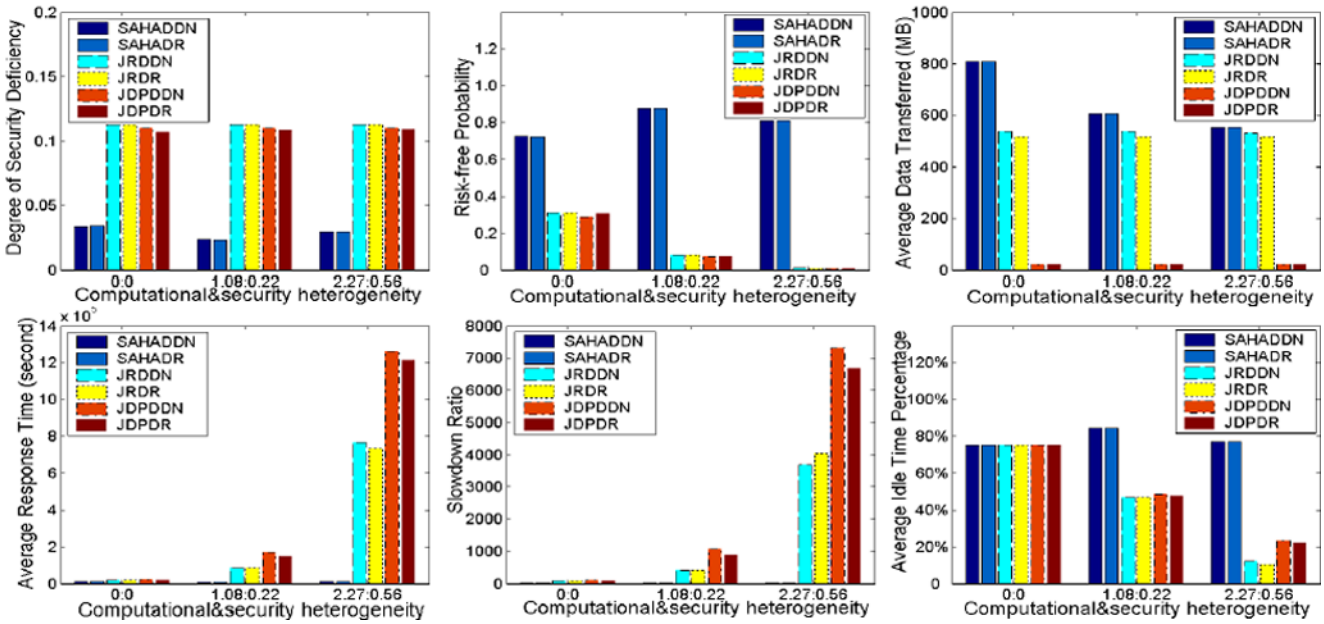


Fig. 4 Performance impact of computational and security heterogeneity

putational and security overhead. In summary, the strength of SAHA can be fully demonstrated when a Grid is heterogenous in both computation and security, which is a common scenario in real world. The results suggest that SAHA can be successfully applied in an existing security-critical data Grid.

5.4 Impact of characteristics of datasets

In this experiment we evaluate the performance impact of features of datasets required by the entire job set. The features of datasets include distribution of dataset popularity, threshold of dataset popularity, number of sites for each dataset, and number of datasets (see Table 1).

The popularity of a given dataset is defined as the total number of requests issued by the entire job set on the dataset. Since the distribution of dataset popularity influence the system performance, we tested three distributions of dataset popularity, i.e., uniform distribution, normal distribution, and geometric distribution in this experiment. All other experiments use a local uniform distribution of dataset popularity. The uniform distribution resembles an ideal case where each job randomly selects a dataset to access. The normal distribution reflects the fact that a majority of jobs only request a subset of datasets. The geometric distribution models the scenario in which a group of users interests on some datasets more than others.

The experimental results are shown in Fig. 5. There are several important observations based on Fig. 5. Firstly, SAHA significantly outperforms the other two scheduling algorithms in security, average response time, and slowdown ratio in all the three distribution cases. Moreover, the distribution of dataset popularity has no impact on security performance. Secondly, the normal distribution leads to the shortest average response time and slowdown ratio. This is because the requests on the datasets are evenly distributed, and thus, the workload is well balanced. Thirdly, the geometric distribution results in the lowest volume of average data transferred. This can be explained by the fact that more dataset replicas of a particular dataset are created during the course of job scheduling for the dataset is frequently requested, which is a characteristic decided by geometric distribution. Thus, the subsequent jobs that issue requests on the dataset are most likely to be assigned to a site where a replica of the dataset was already there. Therefore, no data transmission is needed.

The DataRandom data replication policy keeps track of the popularity of each dataset. When the popularity of a dataset exceeds a threshold value, a replica of the dataset is replicated to a random site on the Grid [7]. To evaluate the impact of the threshold value, we conducted the following experiment (see Fig. 6).

Figure 6 reveals that the value of dataset popularity threshold has no influence on security. However, it does affect system performance in average response time, slowdown ratio, and average data transferred. The lower value of the threshold is, the more replicas are created. Thus, the average response time and the slowdown ratio of JDPDR increase when the threshold is escalating. However, SAHADR almost keeps constant in these metrics as the threshold value increases. This is because SAHADR take into account the

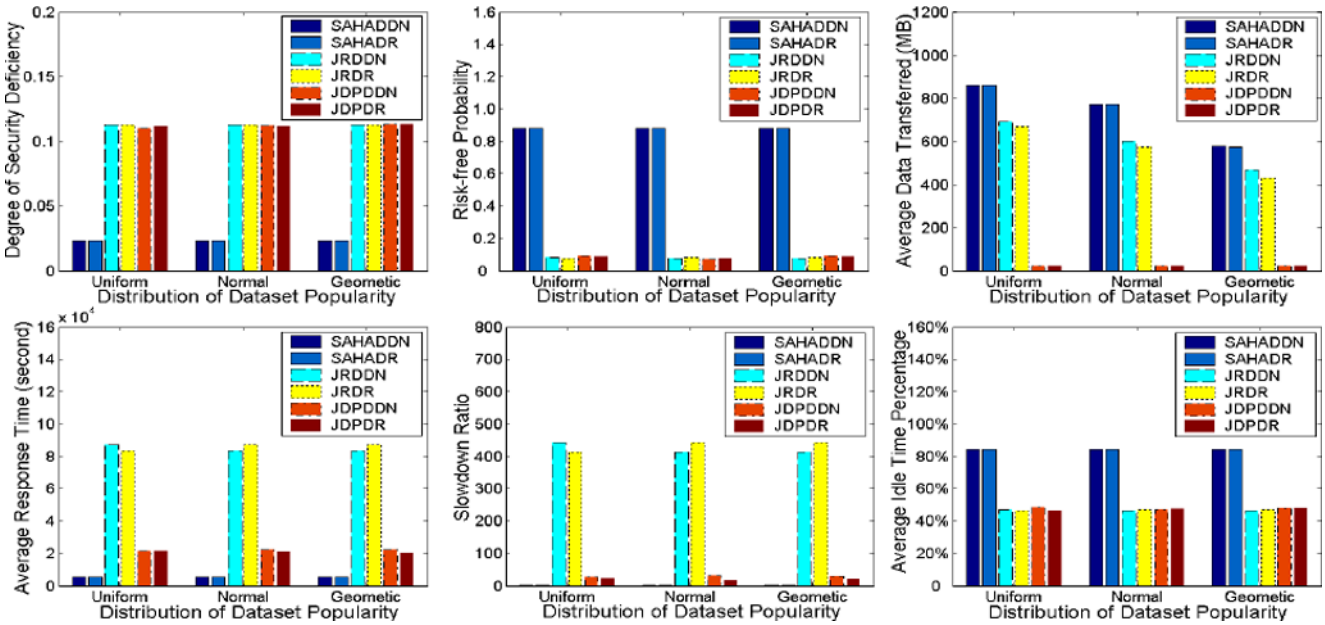


Fig. 5 Performance impact of distribution of dataset popularity

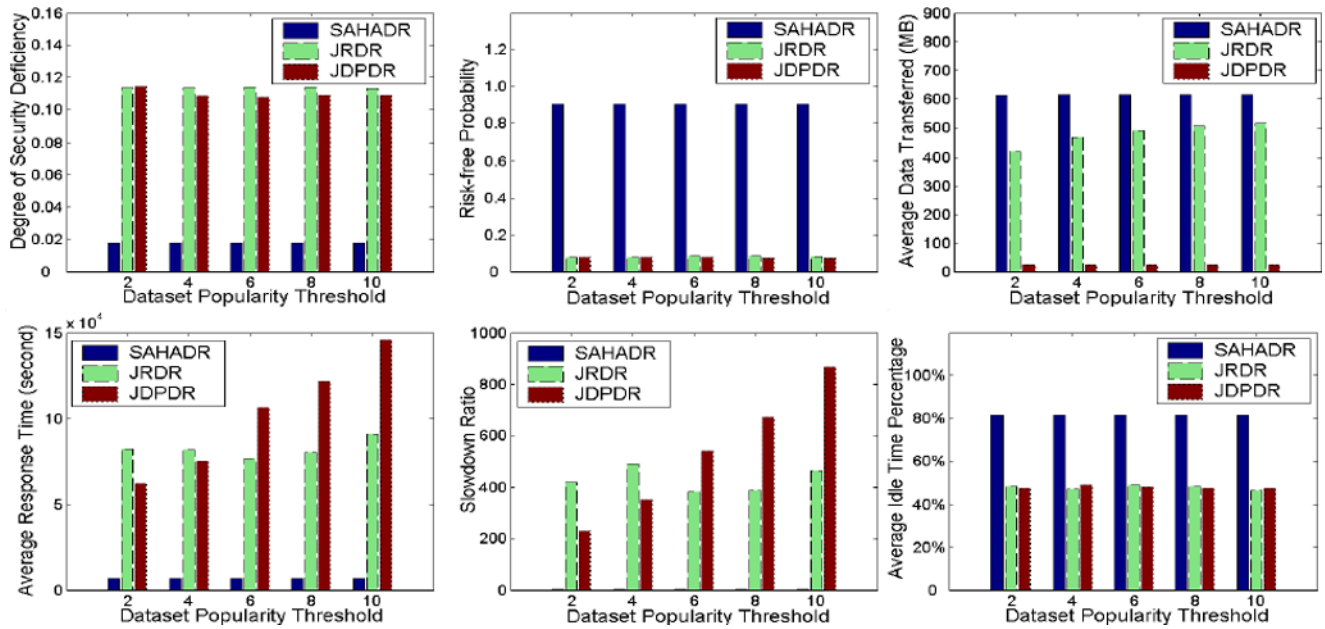


Fig. 6 Performance impact of dataset popularity threshold

computational time and security overhead heterogeneities, which are two dominant items in the total response time of a job. Compared with these two items, the data transmission time item is trivial. That is why the dataset popularity threshold cannot obviously affect the performance in the metrics mentioned above. As for JRDR algorithm, it exhibits some extent randomness in average response time and slowdown ratio. This is decided by the random nature of the scheduling algorithm, which randomly selects a site for a job.

6 Summary and future work

In this paper, a novel security- and heterogeneity-driven scheduling strategy was developed for data-intensive applications on data grids. In the first part of this study, we constructed a model for data-intensive jobs with security requirements. In addition, we proposed a new security overhead model to measure security overheads experienced by data-intensive jobs. Our scheduling strategy takes into account datasets accessed and generated by applications,

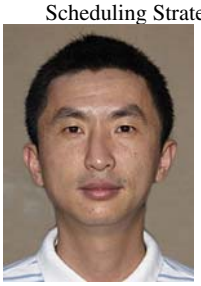
thereby maintaining high performance for data grids. By using the new concept of degree of security deficiency, this scheduling strategy is capable of improving quality of security for data grids. We implemented a trace-driven simulator to evaluate the performance of our approach under a wide range of workload characteristics. Compared with four existing scheduling algorithms, our strategy achieved improvements in security and performance by up to 810% and 1478%, respectively.

We qualitatively assigned security levels to the security services based on their respective security capacities. In our future work, we intend to investigate a quantitative way of reasonably specifying the security level of each security mechanism.

Acknowledgements This work was partially supported by a research grant 2005-04-070 from the Intel Corporation and a research fund (103295) from the New Mexico Institute of Mining and Technology.

References

1. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., Tuecke, S.: The data grid: towards an architecture for the distributed management and analysis of large scientific datasets. J. Netw. Comput. Appl. **23**, 187–200 (2001)
2. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: enabling scalable virtual organizations. Int. Journal Supercomput. Appl. **15**(3), 200–222 (2001)
3. Keahey, K., Welch, V.: Fine-grain authorization for resource management in the grid environment. In: Proc. Int'l Workshop Grid Computing, 2002
4. Novotny, J., Tuecke, S., Welch, V.: An online credential repository for the grid: MyProxy. In: Proc. Int'l Symp. High Performance Distributed Computing, August 2001
5. Park, S.-M., Kim, J.-H.: Chameleon: a resource scheduler in a data grid environment. In: Proc. Int'l Symp. Cluster Computing and the Grid, 2003
6. Qin, X., Jiang, H.: Data grids: supporting data-intensive applications in wide area networks. In: Yang, L., Guo, M.-Y. (eds.) High Performance Computing: Paradigm and Infrastructure, Wiley, Hoboken (2004)
7. Ranganathan, K., Foster, I.: Decoupling computation and data scheduling in distributed data-intensive applications. In: Proc. IEEE Int. Symp. High Performance Distributed Computing, 2002, pp. 352–358
8. Song, S., Kwok, Y.-K., Hwang, K.: Trusted job scheduling in open computational grids: security-driven heuristics and a fast genetic algorithms. In: Proc. Int'l Symp. Parallel and Distributed Processing, 2005
9. Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, L., Tuecke, S.: Security for grid services. In: Proc. Int'l Symp. High Performance Distr. Computing, 2003
10. Winton, L.: Data grids and high energy physics: a Melbourne perspective. Space Sci. Rev. **107**(1–2), 523–540 (2003)
11. Xie, T., Qin, X., Sung, A.: SAREC: a security-aware scheduling strategy for real-time applications on clusters. In: Proc. 34th Int'l Conf. Parallel Processing, Norway, June 2005
12. Xie, T., Qin, X.: Enhancing security of real-time applications on grids through dynamic scheduling. In: Proc. 11th Workshop Job Scheduling Strategies for Parallel Processing, MA, June 2005



Tao Xie received the PhD degree in computer science from the New Mexico Institute of Mining and Technology in 2006. He received the BSc and MSc degrees from Hefei University of Technology, China, in 1991 and 2000, respectively. He is currently an assistant professor in the Department of Computer Science at San Diego State University, San Diego, California. His research interests are security-aware scheduling, high performance computing, cluster and Grid computing, parallel and distributed systems, real-time/embedded systems, and information security. He is a member of the IEEE.

Xiao Qin received the BSc and MSc degrees in computer science from Huazhong University of Science and Technology, China, in 1996 and 1999, respectively. He received the PhD degree in computer science from the University of Nebraska-Lincoln in 2004. He is an assistant professor in the Department of Computer Science at the New Mexico Institute of Mining and Technology. His research interests are parallel/distributed systems, real-time computing, storage systems, and performance evaluation. He has published more than 50 technical papers and served as program committee of several prestigious conferences, including the 35th International Conference on Parallel Processing. He is a member of the IEEE.