Trusted Grid Computing with Security Assurance and Resource Optimization

Shanshan Song and Kai Hwang
University of Southern California
Los Angeles, CA. 90089 USA
{shanshas, kaihwang}@usc.edu

Abstract

The security of Grid sites can be enhanced by upgrading its intrusion defense capabilities against its previous job success rate on Grid platforms. A new fuzzy-logic trust model is proposed for distributed security enforcement across multiple Grid resources sites. The design is aimed at securing Grid resources with optimized resources subject to budget constraints. The performance of trusted Grid computing is verified by simulated trust integration over multiple Grid resource sites.

The SARAH scheme scales well with increasing number of divisible user jobs and can sustain high efficiency, as more resource sites are added. Integrated trust and resource optimization make it possible to accommodate all user applications with low job drops and short waiting time. As a result, the Grid resources are better utilized for distributed execution of large number of user jobs.

Keywords: **Grid computing, trust management, fuzzy logic, resource allocation, nonlinear programming, and distributed Computing**

1. Introduction

In Grid computing systems, security protection and resource allocation are two basic problems to solve [6]. User programs may contain malicious codes that may endanger the Grid resources. Shared Grid resources once infected by malicious codes may damage user applications running on the Grid platforms. We address these issues by allocating Grid resources with security assurance and precautions to avoid application disasters in an open Grid environment.

Highly-shared Grid resources create severe insecurity and privacy concerns, that have hindered the acceptance of distributed Grid applications [12]. This paper presents a Grid resource management architecture based on highly secured allocation of resources to user applications. The scheme satisfies the demand of computing power and security assurance under certain budget constraints. We aim to avoid Grid failures from platform vulnerability or malicious attacks. The purpose is to achieve the highest performance at low cost.

We have checked the security and privacy requirements in reported Grid applications [12]. These Grid applications cover scientific explorations, health-care, public safety, national security, government and business services, etc. For an example in the health-care area, the Grid offers medical records needed to perform remote diagnosis, and medical treatment [2]. Global-scale medical data Grids are thus needed. The e-Science at UK, VegaGrid in China, and MammoGrids in Europe offer these applications [2].

The trustworthiness of all resource sites allocated must be established in Grid computing. The trust was classified into *identity trust* and *behavior trust* [1]. The identity trust is concerned with the authentication of clients and servers, and resource access control through authorization. The Globus *Grid Security Infrastructure* (GSI) was developed to provide these services [2]. The behavior trust deals with an entity's trustworthiness.

In this study, a fuzzy logic trust model is proposed for modeling behavior trust among resource sites in a Grid computing system. We try to quantify the trustworthiness of resource sites, based on previous job execution experiences and assessed by self-defense capabilities of resource sites. In the rest of paper, trust is equivalent to behavior trust unless explicitly specified. Our trust management is meant to enforce security in Grids with security-assured resource allocation.

Like human relationship, trust is expressed by a linguistics term rather numerically. Fuzzy logic is very suitable to quantify the behavior trust among peer groups. The fuzzy theory has not been exploited much for security control in distributed computing. We offer the first attempt to manage trust with fuzzy inference in Grid computing applications [10].

Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems **(PDCS-2004), September 15-17, 2004, San Francisco, CA, USA. This research work was supported by a NSF ITR Grant ACI-0325409.**

Two advantages of using fuzzy-logic for trust management in Grid computing are identified below: (1) Fuzzy inference is capable of quantifying the imprecise data or uncertainty involved in assessing the security of resource sites. (2) Different inference rules could be developed for different Grid applications using the same fuzzy-logic inference engine. These two advantages are demonstrated in this paper.

In our schema, we consider $\boldsymbol{m}$ resource sites. Each site is characterized by a triplet $\boldsymbol{R_i} = (\boldsymbol{P_i}, \boldsymbol{V_i}, \boldsymbol{C_i})$, representing the *computing power*, *trust vector*, and *unit service cost* at the $\boldsymbol{i}$-th resource site. We reserve the index $1 \le \boldsymbol{j} \le \boldsymbol{m}$ to denote the resource site evaluating other sites and the index $1 \le \boldsymbol{i} \le \boldsymbol{m}$ for the sites being evaluated. The mutual trust indices among resource sites are modeled by a *trust matrix* $\boldsymbol{M}$. The *job success rate* Φ, *intrusion defense capability* Δ, and *trust index* Γ are properties of individual site.

A large-scale Grid application can be subdivided into many small jobs for parallel execution on distributed Grid sites [5]. In this paper, we concern only the secure resource allocation to subdivided jobs. The job partitioning and program flow analysis are beyond the scope of our study.

All communication overheads are merged into job workloads. We assume reliable network links between Grid sites. For a given job, only a fraction $\boldsymbol{x_i}$ ($0 \le \boldsymbol{x_i} \le 1$) of a site's computing power is drafted for executing the job. One resource site could be utilized for executing a series of jobs simultaneously.

A subdivided job is characterized by a 4-tuple $\boldsymbol{Job} = (\boldsymbol{W}, \boldsymbol{D}, \boldsymbol{T}, \boldsymbol{B})$, representing the *workload*, *execution deadline*, *minimum trust requirement*, and *budget limit* associated with the execution of the job. Based on the fuzzy-trust model, a SARAH (*Security-Assured Resource Allocation architecture*) scheme is proposed for resource co-allocation on divisible workload in the Grid applications.

The remaining sections are organized as follows. In Section 2, the SARAH scheme is presented. Section 3 presents the fuzzy-logic trust model for Grid security management. Section 4 presents the trust update and propagation algorithms across multiple Grid sites.

Section 5 specifies the nonlinear optimization model for secure Grid resource allocation. Section 6 reports the simulation results on trusted resource allocation using the SARAH scheme. Finally, we summarize the research findings, lessons learned, and suggest further research directions.

2. Security Assurence in Grid Computing

The Grid service providers must assure the users with guaranteed security, privacy protection, and dependable accessibility of all Grid-enabling platforms [7], [8]. We envision a highly-secure Grid environment, in which server sites, Intranets, or Grid resources are well protected. Figure 1 shows the SARAH architecture for security assurance and optimized resource management.

We establish a minimum number of encrypted VPN channels among Grid sites. The trust management is done directly over the VPNs. Using encrypted channels in a Grid reduces or eliminates the overheads in frequent authentication, trust propagation, key management, and authorization in most Grid operations. For individuals or small business, the PKI-based Grid services may be more economical. For enterprises or government Grid services, we recommend the construction of dedicated VPN with encrypted channels on top of the PKI certificates.

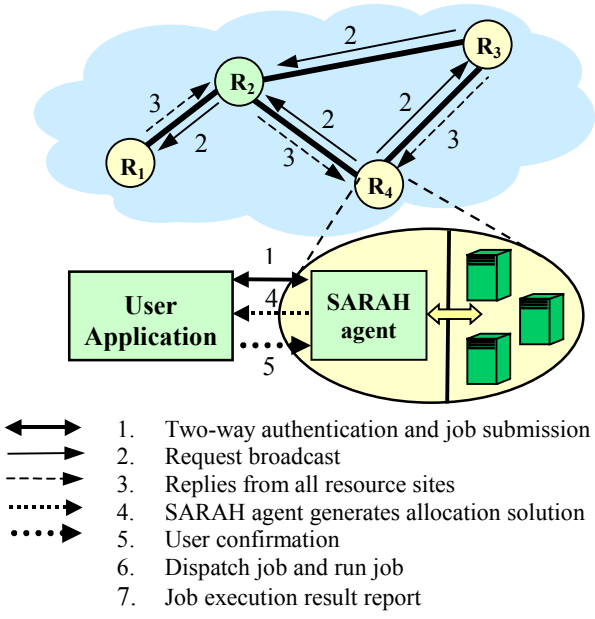


Fig. 1. Trusted Grid resource allocation in USC GridSec project and seven steps in allocating secure resources to provide user applications with security assurance and sustained performance/cost ratio

We avoid the use of a single resource scheduler. The security-assurance steps are shown in Fig. 1 over four sites $\{\boldsymbol{R_1}, \boldsymbol{R_2}, \boldsymbol{R_3}, \boldsymbol{R_4}\}$. The user submits request to the SARAH agent in site $\boldsymbol{R_4}$. Two-way authentication is done first between user application and the agent.

The demanded computing power and available budget are provided during the submission. The agent broadcasts the request to agents in other sites. The SARAH agents work collectively to generate the optimal allocation solutions.

In Fig. 1, the user application chooses three resources $\{\boldsymbol{R_1}, \boldsymbol{R_3}, \boldsymbol{R_4}\}$, based on computing power, security demand, and budget constraints. The VPN connections are established between three resource sites for data transfer

and communication purposes. Steps 2 through 5 could be repeated, if the user is not satisfied with any of the solutions. The SARAH is implemented with security assurance through trust integration as described in Section 5. The SARAH agents communicate with each other to schedule jobs and manage resources.

3. Fuzzy Logic for Trust Management

We consider m resource sites. Each site is characterized by the 3-tuple $R_i = (P_i, V_i, C_i)$, defined in Table 1. Previous job execution experiences determine the trustworthiness of resource sites. Each resource site R_j maintains a *trust vector* defined by a column vector: $V_j = (t_{1j}, t_{2j}, \ldots, t_{mj})^T$.

The *trust index* t_{ij} for $1 \le i, j \le m$ represents the trust of site R_i by site R_j. The *trust index* t_{ij}, of a site R_j is a fraction in the range [0, 1], where 0 represents the most risky case with no protection and 1 for a total trusted condition with the highest level of security assurance.

Being asymmetric, t_{ij} is not necessarily equal to t_{ji}. Any value in between indicates a partially secured resource site. Trust vector V_j is used to allocate resources to jobs submitted to R_j. Combining all trust vectors, we define a *trust matrix* for all m Grid resource sites as: $M = (V_1, V_2, \ldots, V_m) = (t_{ij})_{m \times m}$.

The trust relationships among Grid sites are hard to assess due to uncertainties involved. Fuzzy inference is capable of quantifying imprecise data or uncertainty associated with security index of resource sites. Security holes may appear as OS blind spots, software bugs, privacy traps, and hardware weakness in resource sites. These holes may weaken the trust index value.

In our SARAH scheme, the *trust index* Γ (or t_{ij}) is determined by the job *success rate* Φ and intrusion *defense capability* Δ of a pair of resource sites. Our trust model is built with fuzzy inference of the trust indices at individual resource sites. Incremental trust update and trust propagation are periodically performed over all sites.

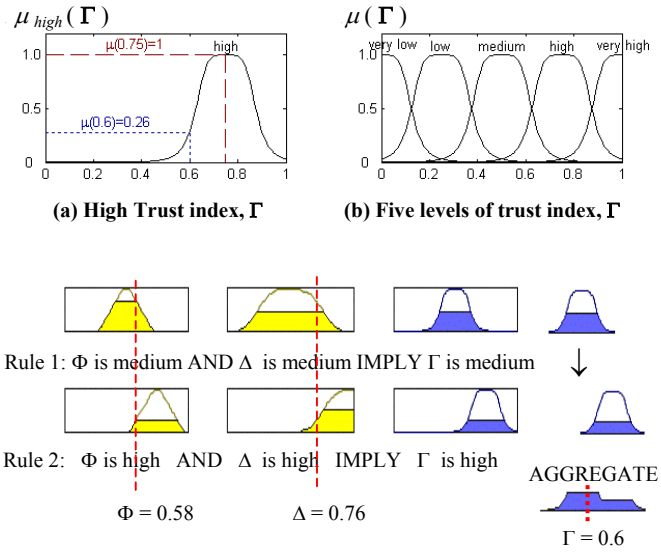
In Fuzzy logic, the *membership function* $\mu(x)$ for a fuzzy element x in a fuzzy set specifies the degree to which an element belongs to that fuzzy set. It maps element x into the interval [0, 1], while 1 for full membership and 0 for no membership.

Figure 2(a) shows "high" membership function for trust index Γ. A resource site with $\Gamma = 0.75$ is considered full membership, while $\Gamma = 0.6$ corresponds to a lower trust index. Figure 2(b) shows five membership functions corresponding to *very low*, *low*, *medium*, *high*, and *very high* degree of trustworthiness.

Fuzzy inference is a process to assess the trust index in five steps: (1) Register the initial values of the success rate Φ and defense capability Δ. (2) Use the membership functions to generate membership degrees for Φ and Δ. (3) Apply the fuzzy rule set to map the input space (Φ - Δ space) onto the output space (Γ space) through fuzzy 'AND' and 'IMPLY' operations. (4) Aggregate the outputs from each rules, and (5) Derive the trust index through a defuzzification process. The details of these five steps can be found in reference [9].

Figure 2 (c) shows the trust inference process using the membership functions in Fig.2 (a) and (b). We consider initial values: $\Phi = 0.58$ and $\Delta = 0.76$, obtained from previous Grid application experiences. Two example rules are given below for illustrating the inference process.

Rule 1: If Φ is medium and Δ is medium, then Γ is medium.
Rule 2: If Φ is high and Δ is high, then Γ is high.



(a) High Trust index, Γ **(b) Five levels of trust index, Γ**

(c) Fuzzy inference procedure

Fig. 2. Membership functions for trust index, and fuzzy logic inference for trust integration

All selected rules are inferred in parallel. We explain below the details of applying rule 1. Initially, the degrees of membership are determined for all terms in the premise of rule 1. The fuzzy operator 'AND' is applied to determine the support degree of rule 1. Furthermore, the degree of support for this rule is used to shape the output set of consequences ('Γ is medium' in rule 1).

Both outputs from rule 1 and rule 2 are aggregated together. The final trust index $\Gamma = 0.6$ is generated by defuzzifying the aggregation. The "AGGREGATE" operation is done by superimposing the two resulting curves at the lower right corner of Fig. 2 (c).

The trust index varies with respect to the job success rate and self-defense capability. The trust index increases with the increase of both contributing factors. We will

introduce a trust integration process in the next section to guide the site fortification or security enhancement at all resources sites. All the resource sites are involved in this collective security upgrading process.

4. Trust Integration across Resource Sites

The concept of fuzzy trust integration was first introduced in our earlier report [10]. Suppose that site R_i has monitored all jobs submitted to R_i and executed on site R_j for some time to know R_j's job execution success rate and defense capability. Let s_{ij} be the new security stimulus between sites R_i and R_j given at certain time instant. Equation (3) is used to calculate the new trust index from the old trust index and the present stimulus value.

$$t_{ij}^{new} = \alpha t_{ij}^{old} + (1-\alpha)s_{ij} \quad (1)$$

The weighting factor α is a random variable in the range (0,1). For security critical applications, the trust index should change quickly to reflect new situation, thus a small α is adopted such as $\alpha < 0.3$. But for the stable and relatively low security-sensitive applications, a large α is adopted such as $\alpha > 0.9$. In general, situations, one can set α in the range of (0.7, 0.8).

SARAH agents broadcast trust vectors to each other periodically. The broadcast timeout depends on specific Grid applications. For the highly security sensitive applications, shorter timeout period should be adopted. Otherwise, longer update period could be used. With m resource sites, the contribution from each site is roughly $1/m$. We use the following Eq. (2) to calculate new trust vector for site R_i, considering the impact from another resource site R_j.

$$V_i^{new} = \frac{m-1}{m} V_i^{old} + \frac{1}{m} V_j \quad (2)$$

Our trust integration scheme periodically monitors the security indices of all Grid resource sites. When the defense capability of a resource site is assessed low, the scheme demands the site to upgrade its defense by hardware fortification, software installations, or security policy changes.

Typical site defensive measures could include installation of smart packet filters, stateful firewalls, traffic monitors, and IDS (*intrusion detection system*), etc. The aim is to reduce the site vulnerability and by upgrading its defense capability Δ. These security enhancements are meant to block network attacks or to enforce fine-grain access control on critical resources.

We generate below this defense upgrade after each trust integration step.

$$\Delta = \Delta + \varepsilon(\Delta) \quad (3)$$

The increment ε is a function of the current Δ value. If current Δ is high, ε should be set with a small increment. If Δ is low, ε should be larger. The site that has low Δ should catch up faster this way. The ultimate purpose of the trust integration process to enhance the trust index or security level at all weak resource sites.

Thus, the integration leads to the effects of normalization and equalization of trust indices. In our simulation experiments, we vary the Δ to elevate the trust indices at all weak resource sites. Figure 3 shows a trust integration example of four resource sites, characterized by a 4×4 trust matrix.

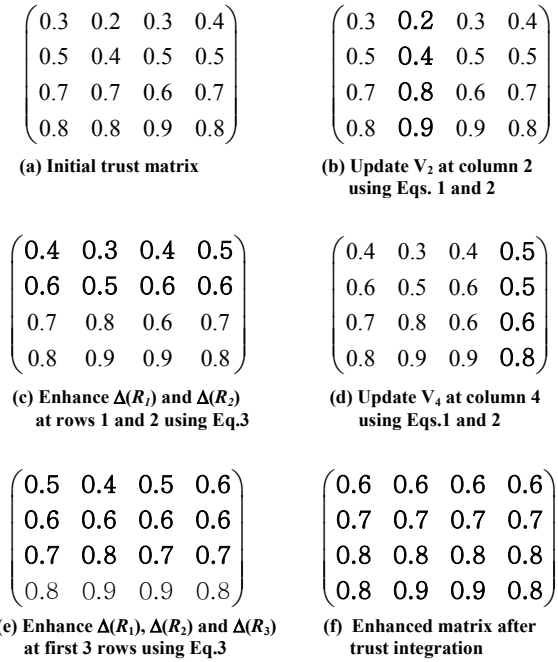


Fig. 3. Snap shots of the trust integration process over four resource sites. The initial matrix entries have a high disparity. Through a sequence of trust update, trust propagation, and security enhancement operations, the final trust indices of all four sites are upgraded to satisfy the demand from user jobs

The initial values are given in Part (a). Assume $\alpha = 0.5$ and R_2 monitors the success rate and defense capability of R_3 and R_4. Based on fuzzy quantification, the stimulus values $s_{32} = 0.9$ and $s_{42} = 1.0$ are calculated by newly gathered security information. Using Eq. (1), the new trust indices $t_{32} = 0.8$ and $t_{42} = 1.0$ are calculated in Part (b).

At the same time, R_2 detects that the trust indices of R_1 and its own are lower than the user demand. Thus, need to enhance the defense capability of R_1 and R_2. The trust indices of R_1 and R_2 are enhanced in Part (c), accordingly.

Similarly, R_4 monitors the success rate and defense capability of R_3. Thus, the trust vector V_4 is updated in Part (d). The trust indices of R_1 and R_2 and R_3 are all assessed low and they are upgraded in Part (e). Finally, the trust indices in all four sites are upgraded to satisfy the user requirements in Part (f).

5. Optimized Resource Allocation

In this section, we model the Grid resource optimization process as a dual-objective nonlinear programming problem. Each resource site R_i is characterized by a 3-tuple $R_i = (P_i, V_i, C_i)$. The *computing power* P_i of a resource site is decided by the CPU speed, storage capacity, I/O bandwidth etc.

We choose the unit of Tflop/s (*Tera floating-point operations per second*) to measure the computing power of a computational Grid. In case of data Grids, different performance unit could be chosen. Based on the fuzzy trust model, we present below the SARAH scheme for optimized Grid resource-allocation with nonlinear programming.

Algorithm 1: ***SARAH*** **(R_j, *Job* = (W, D, T, B))**
Input: Submit a ***Job*** to site R_j at time τ; R_j requests resources from all m sites to execute this job.
Output: Workload distribution $(W_1, W_2, \ldots, W_m)$ and estimated execution time L for ***Job*** based on allocation $X = (x_1, x_2, \ldots, x_m)$ generated.
(1) R_j sends requests resources from all sites.
(2) **for** $i = 1$ **to** m
(3) **if** $(t_{ij} < T)$ $x_i = 0$.
(4) **end for**
(5) Estimate execution time $L = D - \tau$.
(6) Generate the allocation vector $X = (x_1, x_2, \ldots, x_m)$, which maximize $E = \sum_{i=1}^{m} x_i P_i L t_{ij} \Big/ \sum_{i=1}^{m} x_i P_i L C_i$, subject to the following constraints
$\sum_{i=1}^{m} x_i P_i L \geq W$, $\sum_{i=1}^{m} x_i P_i L C_i \leq B$, and $0 \leq x_i \leq 1$.
(7) **for** $i = 1$ **to** m $W_i = x_i P_i L$
(8) **return** workload distribution and estimated execution time $(W_1, W_2, \ldots, W_m, L)$ with respect to resource allocation vector $X = (x_1, x_2, \ldots, x_m)$.

Denote the current time instant by τ, this is the start time of a job execution. The *estimated job execution time* is denoted by $L = D - \tau$. Let x_i be the percentage of the peak power P_i allocated to a job. The product $W_i = x_i P_i L$ is the *workload* for site R_i.

The SARAH scheme was developed under the following assumptions: (1) non-preemptive job scheduling, (2) divisible workload for parallel processing across Grid sites, and (3) space sharing in a batch mode instead of time-sharing mode over multiple jobs.

The input to Algorithm 1 is the successive job descriptions including the site and time of submission. After passing a qualification test in Steps (2) - (4), the unqualified sites are filtered out. The estimated execution time L is registered at Step 5.

The *resource allocation vector* $X = (x_1, x_2, \ldots, x_m)$ is generated by optimizing the following objective function E, called the *performance/cost ratio*. Essentially, this measure reflects the *efficiency* of the SARAH scheme.

$$E = \sum_{i=1}^{m} W_i t_{ij} \Big/ \sum_{i=1}^{m} W_i C_i = \sum_{i=1}^{m} x_i P_i L t_{ij} \Big/ \sum_{i=1}^{m} x_i P_i L C_i \qquad (4)$$

The above objective function is solved with a nonlinear programming method describe in [4]. Our simulator apply the implemented KNITRO package [11], a solver for large-scale non-linear optimization problems. The solution is subject to those inequality constraints listed in Step 6.

Considering a job submitted to site R_j, this site is responsible for allocating resources from all sites. Site R_j is also responsible for monitoring the job execution from the beginning to the end. Once a job is finished, the allocated resources are released for other jobs.

User applications can resubmit their jobs, if the earlier submission was unsuccessful. The output of Algorithm 1 is the workload distribution and estimated job execution time $(W_1, W_2, \ldots, W_m, L)$ with allocation vector $X = (x_1, x_2, \ldots, x_m)$ to assign resources at m sites for executing a sequence of n jobs.

6. Experimental Results

We have simulated n = 300 to 10,000 jobs running on m = 6 to 100 Grid sites. Each site is modeled by a random computing power in the range of 0.5 – 5 Tflop/s. Each site defense capability is modeled by a random variable in [0, 1]. Job arrivals are modeled by a Poisson distribution.

The minimum trust index of a site is set between 0.4 and 0.6, randomly. Both the resource unit service charge and user application budget limit are randomly set in the range of $280 - $320K/Tflop.

A. Experimental Setup

Figure 4 shows the simulation architecture for a typical Grid resource site R_i. We simulate m such resource sites. Jobs are generated by the *Random Job Generator* and sent to the *Job Queue*. The central component is the SARAH scheduler. To guide the implementation of its decisions, the SARAH scheduler uses *KNITRO solver* [11] and the *Trust Manager*.

The SARAH scheduler gathers information from all Grid sites. All sites have the same configuration. The scheduler sends available resource information to KNITRO solver. This solver returns the allocation solution for a single job. Trust manager maintains trust information, and guide the scheduler's decision making. All results are output to the result file.

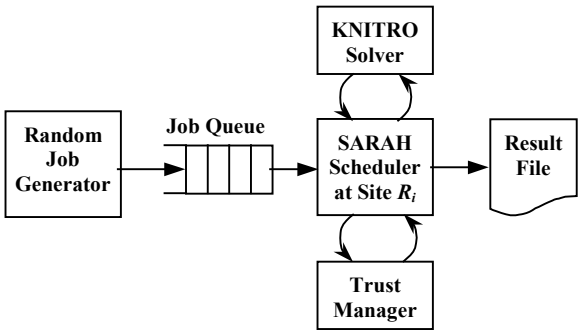
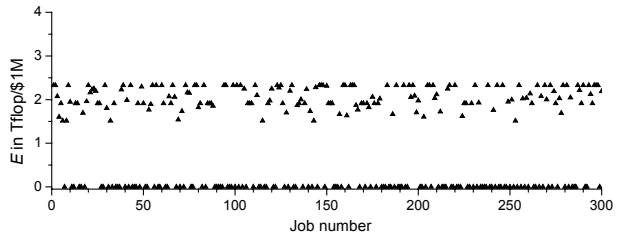


Fig. 4. Simulation architecture for a typical Grid site

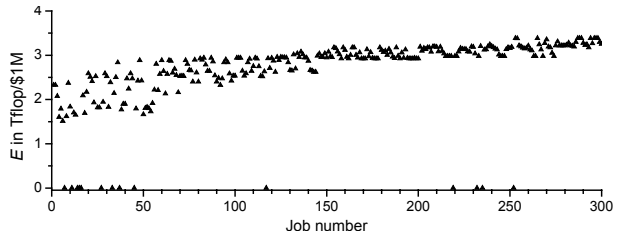
B. Grid Performance/Cost Ratio or Grid Efficiency (*E*)

Three hundred jobs are simulated on six resource sites. Jobs are submitted with an inter-arrival time of 4 minutes. The job workload demand varies from 100 to 5,000 Tflop. The deadline is set 20 - 120 minutes after submission.

We present in Fig.5 two scatter plots of the *performance/cost ratio* ***E*** defined in Eq.(4). The two scatter plots result from simulating the SARAH operations under two different trust management scenarios: *fixed trust* and *integrated trust*. Each triangle represents one job executed on the Grid site. Dropped jobs are shown by triangles lying on the x-axis.



(a) Fixed trust without security upgrade



(b) Integrated trust with security enhancement

Fig. 5. Scatter plots of the performance/cost ratio of two Grid resource allocation scenarios for executing 300 jobs over six resource sites

In Fig.5(a), the *E*-plot for successful jobs varies from 1.5 ~ 2.4 Tflop/$1M with an average E = 2.07 Tflop/$1M. In Fig.5(b), the *E*-plot for successful jobs varies from 1.5 ~ 3.38 Tflop/$1M with an average E = 2.83 Tflop/$1M. The two average values on E show a 37% = (2.83-2.07)/2.07 advantage in favor of trust-integrated resource allocation.

The *E*-plot increases steadily as more jobs submitted in Fig.5(b). Considering the last 50 jobs, we achieved E = 3.0 ~ 3.38 Tflop/$1M. Comparing the last 50 jobs in Fig. 5(a) and (b), we observe a 125% = (3.38 - 1.5)/1.5 improvement in the best-case scenario.

C. Job Drop Rate and Average Waiting Time

Jobs are dropped if the resources are insecure or inadequate to satisfy the job demand. In Fig.5, dropped jobs are shown along the *x*-axis with $\boldsymbol{E} = 0$. The results on *job drop rate* are tabulated in Table 1.

Table 1. Job Drop Rate and Average Waiting Time of 300 Jobs for Parallel Execution on six Grid Sites

Job Number	Number of dropped jobs		Average waiting time in minutes	
	Fixed trust	Trust integration	Fixed trust	Trust integration
1 - 50	20	9	28.33	16.05
51 - 100	22	0	30.67	3.87
101-150	19	1	27.90	2.51
151-200	21	0	35.33	1.00
201-250	28	3	34.11	3.31
251-300	20	1	20.86	1.82

With fixed trust, the number of dropped jobs remains constant around 20 jobs out of every 50 jobs processed. Among 300 jobs simulated, there are 130 dropped jobs for fixed trust and only 14 dropped jobs for integrated trust. This translates to a reduction of job drop rate from 130/300 = 43.3% to 14/300 = 4.7% after the trust integration.

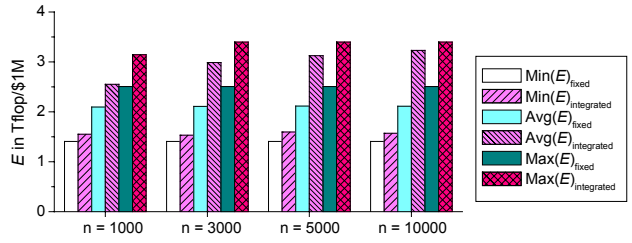
The *average waiting time* is defined as the time to wait for the allocation of resources from the submission of a job. As shown in Table 1, the average waiting time with fixed trust varies between 20 and 36 minutes. The waiting time with trust integration has been significantly reduced from 16 minutes to 1.8 minutes, as more jobs are processed. This shows another advantage of the SARAH scheme.

D. Scalabilty Analysis on Job Number (n)

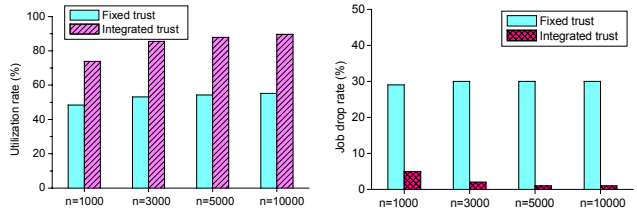
In Fig.6, we study the scalability of our SARAH scheme by simulating the job number $\boldsymbol{n}$ = 1000 to 10000 jobs over $\boldsymbol{m}$ = 20 resource sites. On average, five jobs are submitted to one site within an hour. The maximum job workload demand is enlarged to 12000 Tflop. The deadline is extended to at most 240 minutes.

Fig. 6(a) plots the performance/cost ratio ***E*** against increasing number of simulated jobs. The minimum ***E*** bars for two scenarios are plotted at the left-most pair, the average ***E*** bars at the middle pair, and the maximum ***E*** bars at the right-hand pair. With fixed-trust, the average performance/cost ratio keeps constant at 2.1Tflop/$1M.

On the contrary, the average ***E*** value increases sharply from 2.55 to 3.23 Tflop/$1M in our integrated-trust approach. The maximum ***E*** with trust integration also increases to 3.4Tflop/$1M, much higher than the 2.5Tflop/$1M experienced in fixed-trust case.



(a) Performance/cost ratio *E*



(b) Resource utilization rate (c) Job drop rate

Fig. 6. Scalability of the SARAH scheme against the number of user jobs competing for resources in a computing Grid over 20 sites

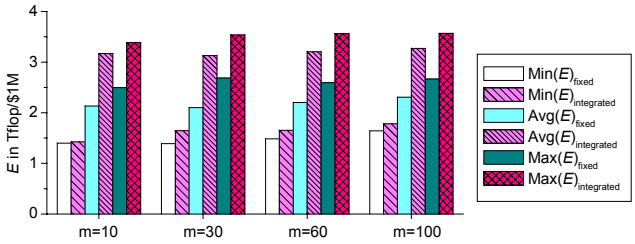
The *utilization rate* is defined as the percentage of allocated resources among all available resources. In Fig.6 (b), the utilization rate of resources with fixed trust values remains at a constant level around 50% during the simulation runs. The utilization of resources with integrated trust varies from a low of 73% to a high of 90%.

The results for job drop rate are shown in Fig. 6(c). Around 30% jobs are dropped for fixed trust scenario while only 1% to at most 5% jobs are dropped for integrated trust scenario. These results demonstrate a significant gain in Grid performance through optimized resource allocation and aggressive security reinforcement.

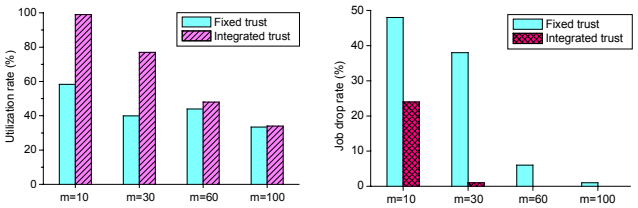
E. Scalability Analysis of Grid Size (m)

In Fig.7, we demonstrate the scalability of our SARAH scheme with respect to the scaling of the Grid configuration from ***m*** = 10 to 100 sites. 5,000 jobs are simulated in these experiments. On the average, 5 jobs are submitted to the same site per hour. The maximum job workload demand is randomly set from 100 Tflop to 12,000 Tflop. The deadline was set 20 minutes to 4 hours after job submission. Other parameters remain constant by default. Only the Gird size is allowed to vary in this study.

The performance/cost ratio ***E*** is plotted in Fig.7(a). For the average ***E*** value, 2.2 Tflop/$1M was observed for fixed-trust resource allocation, while 3.2 Tflop/$1M was observed for integrated-trust resource allocation. This translates to an improvement factor of 45.5% = (3.2 - 2.2)/2.2. The message being conveyed here is that the SARAH scheme results in a higher performance/cost ratio, when more resource sites are added.



(a) Performance/cost ratio *E*



(b) Resource utilization rate (c) Job drop rate

Fig. 7. Scalability of the SARAH scheme in executing 5,000 jobs on m = 10, 30, 60 and 100 resource sites

With fixed 5,000 jobs, the resource utilization decreases in both cases in Fig. 7(b). With trust integration, the utilization rate is 100% for 10 sites, but it reduces to 30% when 100 sites are used. One plausible argument for this is that jobs are not enough to keep all sites busy. As predicted, the job drop rate scales well with adding more resources suites as shown in Fig. 7(c).

F. Summary of Simulated Performance Results

In summary, our trusted resource allocation shows an average of 45% improvement in Grid efficiency *E* over the case of fixed trust without security upgrade. The results clearly demonstrate the effectiveness of trust integration. Integrated trust and resource optimization make it possible to accommodate almost all user applications and very short waiting time before the deadline expires. As a result, the Grid resources are better utilized for job executions.

The results in Fig.6 show that the SARAH resource allocation scheme scales well with increasing number of

jobs. The results in Fig.7 show a sustained higher efficiency *E* as more resource sites are added. As predicted, the utilization rate is low if too many sites are used for handling a small number of user jobs. Overall, we claim that the SARAH scheme outperforms the conventional resource allocation with fixed trust conditions at local resource sites.

7. Lessons Learned and Further Research

We summarize below lessons learned and suggest further research challenges.

A. ***Fuzzy trust integration reduces the platform vulnerability:*** Our fuzzy trust integration guides the defense deployment across distributed Grid sites. This lays the foundation of distributed security enforcement in Grids. The experimental results demonstrate the effectiveness of the fuzzy model for trust integration.

B. ***Trusted Grid resources allocation and configuration*****:** The SARAH model is designed for dynamic resource allocation to meet both computing and security demands. The SARAH scheme appeals to space sharing or batch processing Grid applications. SARAH can be applied to upgrade the AppLeS [2] and NimRod/G schedulers [3] in security reinforcement.

C. ***Benefiting many security-sensitive Grid applications*****:** Grid applications demand not only trusted resources, but also secure communications, multicast authentication, and coherent security policies. Our work will benefit Grid applications in scientific explorations, health-care, public safety, national security, digital government, etc.

D. ***NetShied, a self-defense software library for protecting Grid sites*****:** This is a Grid security enforcement package currently under development at USC. Internet traces and traffic datamining are used to develop automated intrusion detection and response systems to counter the DDoS and flooding attacks.

References:

[1] F. Azzedin and M. Maheswaran. "Evolving and Managing Trust in Grid Computing Systems". *Proc. of the IEEE Canadian Conf. on Electrical Computing Engineering*, 2002.

[2] F. Berman, G. Fox, and T. Hey, (Editors), *Grid Computing: Making The Global Infrastructure a Reality*, John Wiley & Sons, 2003.

[3] R. Buyya, D. Abramason, and J. Giddy, "Nimrod/G: an Architecture for a resource management and scheduling system in a global computational Grid", *Int'l Conf. on High Performance Computing,* 2000.

[4] R. Byrd, M. Hribar, and J. Nocedal. "An Interior Point Method for Large Scale Nonlinear Programming". *SIAM Journal on Optimization*, Vol. 9, no. 4, 1999, pp.877-900.

[5] K. Czajkowski, I. Foster, and C. Kesselman, "Resource Co-Allocation in Computational Grids", *Proc. IEEE Int'l Symp. on High Performance Distributed Computing,* 1999.

[6] M. Humphrey and M. R. Thompson, "Security Implications of Typical Grid Computing Usage Scenarios", *Proc. of Int'l Symp .on High Perf. Distributed Computing* (HPDC), 2001.

[7] J. In, P. Avery, R. Cavanaygh, and S. Ranka, "Policy-Based Scheduling for Simple Quality of Service in Grid Computing", *Proc. of IPPDS 2004*, Santa Fee, April 2004.

[8] C. Liu, L. Yang, I. Foster, and D. Angulo, "Design and Evaluation of a Resource Selection Framework for Grid Applications", *Proc. of Int'l Symp. on High Performance Distributed Computing* (HPDC-11), 2002.

[9] B. Kosko, *Fuzzy Engineering*, Prentice Hall, 1997.

[10] S. Song, K. Hwang, and M. Macwan, "Fuzzy Trust Integration for Security Enforcement in Grid Computing*", in Proceedings of IFIP International Symposium on Network and Parallel Computing,* NPC 2004.

[11] R. Waltz and J. Nocedal, "*KNITRO User's Manual*" Technical Report OTC 2003/05, Optimization Technology Center, Northwestern University, Evanston, IL, USA, April 2003.

[12] V. Welch, F. Siebenlist, I. Foster, et. al. "Security for Grid Services". *In Proceedings of the* 12th *Int'l Symposium on High Performance Distributed Computing* (HPDC-12).

Biographical Sketches:

Shanshan Song received her B.S. degree in Computer Science from the Special Class for Gifted Young in the University of Science and Technology of China in July 2001. She is currently pursuing the Ph.D. degree in Department of Computer Science at the University of Southern California. Her research interest lies primarily in the area of network security and dynamic resource allocation for computational Grids.

Kai Hwang is a Professor and Director of Internet and Grid Computing Laboratory at the University of Southern California. He received the Ph.D. from the University of California, Berkeley. An IEEE Fellow, he specializes in computer architecture, parallel processing, Internet and wireless security, and distributed computing. Presently, he leads a USC research group developing the NetShield/GridSec systems for trusted Grid computing with automated intrusion detection and responses. Visit the GridSec project web site for details of the research tasks performed: http://GridSec.usc.edu/.