Fuzzy Trust Integration for Security Enforcement in Grid Computing*

Shanshan Song, Kai Hwang, and Mikin Macwan

Internet and Grid Computing Laboratory
University of Southern California, Los Angeles, CA. 90089 USA
{shanshas, kaihwang}@usc.edu

Abstract. How to build the mutual trust among Grid resources sites is crucial to secure distributed Grid applications. We suggest enhancing the trust index of resource sites by upgrading their intrusion defense capabilities and checking the success rate of jobs running on the platforms. We propose a new *fuzzy-logic trust model* for securing Grid resources. Grid security is enforced through trust update, propagation, and integration across sites. Fuzzy trust integration reduces platform vulnerability and guides the defense deployment across Grid sites. We developed a SeGO scheduler for trusted Grid resource allocation.

The SeGO scheduler optimizes the aggregate computing power with security assurance under fixed budget constraints. The effectiveness of the scheme was verified by simulation experiments. Our results show up to 90% enhancement in site security. Compared with no trust integration, our scheme leads to 114% improvement in *Grid performance/cost ratio*. The *job drop rate* reduces by 75%. The *utilization of Grid resources* increased to 92.6% as more jobs are submitted. These results demonstrate significant performance gains through optimized resource allocation and aggressive security reinforcement.

1. Introduction

In Grid computing systems [2], user programs containing malicious codes may endanger the Grid resources used. Shared Grid resources once infected may damage the user applications running on the Grid platforms [8]. We address these issues by allocating Grid resources with security assurance. The assurance is achieved by hardware, software, and system upgrades to avoid application disasters in an open Grid environment.

Mutual trust must be established between all participating resource sites. Like human relationship, trust is often expressed by linguistics terms rather numerically. Fuzzy logic is very suitable to quantify trust among peer groups. The fuzzy theory [10] has not been explored much in network security control. To our best knowledge, only Manchala has suggested a fuzzy trust model for securing E-commerce [12].

* The work was presented in the IFIP *International Symposium on Network and Parallel Computing* (NPC-2004), Wuhan, China, October 18-22, 2004. This research was supported by NSF/ITR Grant ACI-0325409 to the University of Southern California.

Azzedin and Maheswaran [3] and Liu and Shen [11] have developed some security-aware models between resource providers and consumers. Globus GSI uses public key certificates and proxies for trust propagation [7]. We use fuzzy inferences to consolidate security enforcement measures in trusted Grid computing.

Our trust assessment involves the measurement of dependability, security, reliability and performability. Trust level is updated after the Grid successfully executed user jobs. Butt and Fortes, et al [3] protect Grid resources from distrusted applications. We choose a reverse approach by assuring security in the resource pool. Figure 1 shows the interaction between two Grid Resource sites.

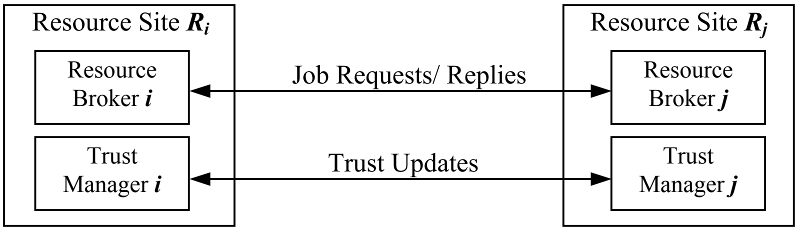

Fig. 1. Securing Grid resources with trust integration and resource brokerage

We propose a *Secure Grid Outsourcing* (SeGO) scheduler to outsource jobs to multiple resources. We aim at maximizing the computing power under security scrutiny and cost minimization. In this paper, we report simulation results on the SeGO performance by executing 300 jobs on six Grid resource sites. Other studies on Grid resource allocation can be found in [2], [4], [6], [14].

The remaining sections are organized as follows. In Section 2, we present our distributed security architecture at USC GridSec project. Section 3 introduces the fuzzy logic for trust management. Section 4 describes the process of fuzzy trust integration. Section 5 introduces the optimized resource allocation scheme. All experimental results are reported in Section 6. Finally, we summarize the research findings and make suggestions for further work.

2. GridSec Project for Trusted Grid Computing

As shown in Fig.1, the GridSec (*Grid Security*) project at USC builds security infrastructure and self-defense toolkits over multiple Grid resource sites. The security functionalities are monitored and coordinated by a *security manager* in each site. All security managers work together to enforce the central security policy. The security managers overlook all resources under their jurisdiction [9]. The GridSec architecture supports scalability, high-security, and system availability in trusted Grid computing. Our purpose is to design for scalability and security assurance at the same time.

V*irtual Private Networks* (VPNs) are built for Grid trust management among private networks through a public network. We establish only a minimum number of encrypted channels in the VPN. The VPN has a number of advantages over the use of PKI in Grid computing. Using encrypted channels in a Grid reduces or eliminates the overheads in frequent authentication; trust propagation, key management, and authorization in most Grid operations [14]. VPN achieves single sign-on easily without using public-key certificates. VPN also reduces the packet exchanges among Grid sites. No certificate authority is needed in VPN-based Grid architecture, once the tunnels are established. The design is aimed at optimizing Grid resources under the constraints of limited computing power, security assurance, and Grid service budget.

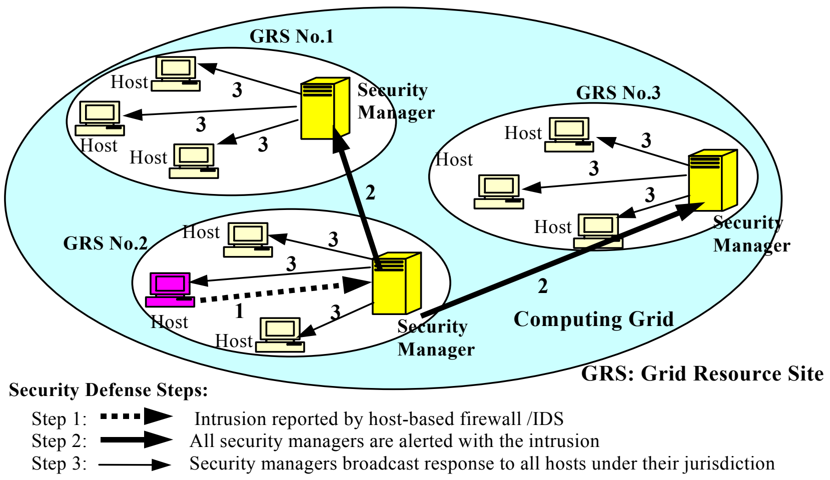
GRS No.1
Host
Host
Host
Security Manager
3
3
3
2
GRS No.3
Host
Host
Host
Security Manager
3
3
3
GRS No.2
Host
Host
Host
Host
Security Manager
3
3
3
1
2
Computing Grid
GRS: Grid Resource Site
Security Defense Steps:
Step 1: Intrusion reported by host-based firewall /IDS
Step 2: All security managers are alerted with the intrusion
Step 3: Security managers broadcast response to all hosts under their jurisdiction

Fig. 2. USC GridSec Project: A distributed Grid security architecture, built with encrypted tunnels, micro firewalls, and hybrid intrusion detection systems, coordinated by cooperative security managers at scattered resource sites

A self-defense software library, called *NetShield,* is under development in the GridSec project. This package supports fine-grain access control with automatic intrusion prevention, detection, and responses [9, 13]. This system is based on dynamic security policies, adaptive cryptographic engines, privacy protection, and VPN tunneling. Dynamic security demands the adaptability in making policy changes at run time. Three steps are shown in Fig.2 for intrusion detection, alert broadcast, and coordinated intrusion responses.

3. Fuzzy Logic for Trust Management

The trust relationships among Grid sites are hard to assess due to uncertainties involved. Two advantages of using fuzzy-logic to quantify trust in Grid applications are: (1) Fuzzy inference is capable of quantifying imprecise data or uncertainty in measuring the security index of resource sites. (2) Different membership functions and inference rules could be developed for different Grid applications, without changing the fuzzy inference engine.

We close up the security gap by mapping only secure resources to Grid applications. 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 scheme, the *trust index* Γ (or t_{ij}) is determined by job *success rate* Φ and self-*defense capability* Δ of each resource pair.

In Fig.3, we plot the variation of the *trust index* of a resource site, as the *job success rate* and site *self-defense capability* are enhanced from low to high values. These two attributes enhance each other on many computer platforms. The trust index increases with the increase of both contributing factors. The trust index could decrease after network attack incidents. This plot guides the trust integration process. We allocate resources with high degree of security assurance. In subsequent sections, we show a systematic method to produce the trusted conditions on Grid sites.

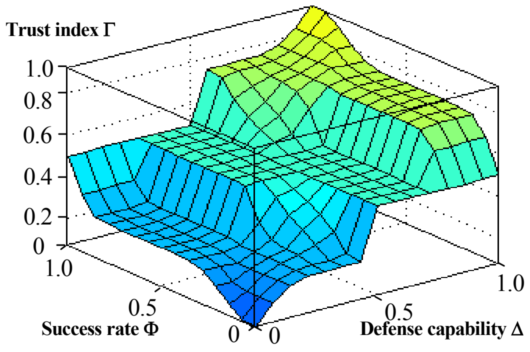

Fig. 3. Variation of trust index Γ with respect to the variations of job success rate Φ and intrusion defense capability Δ at each resource site

Essentially, previous job execution experiences determine the trustworthiness of the peer machines. In the initialization of the trust index of a new resource site, the reported job success rate and intrusion defense capability are used to generate an initial trust value. The trust index is then updated periodically with the site operations, until the site is removed from the Grid domain.

We treat these security attributes as fuzzy variables, characterized by the membership functions in Fig.4. In Fuzzy logic, the *membership function* $\mu(x)$ for a fuzzy element x specifies its degree of membership in a fuzzy set. It maps element x into the interval [0, 1], while 1 for full membership and 0 for no membership. Fuzzy logic can handle imprecise data or uncertainty in the *trust index* of a resource site.

Figure 4(a) shows "high" membership function for trust index Γ. A resource site with 0.75 trust index is considered *high* trust. Figure 4(b) shows five membership functions corresponding to *very low*, *low*, *medium*, *high*, and *very high* degree of trustworthiness. Figure 4(c) shows the cases of three ascending degrees of the self-defense capability. Figure 4(d) shows five levels of job success rate. The inference rules are subject to designer's choice.

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 *Fuzzy Engineering* by Kosko [10].

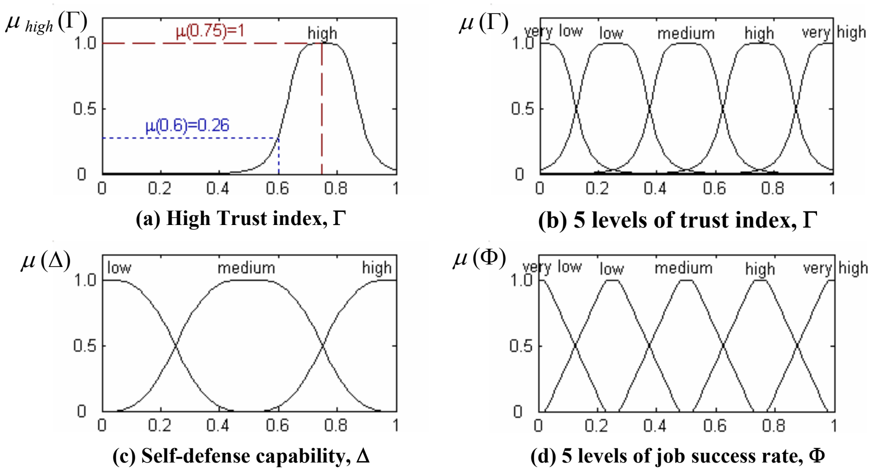

Fig. 4. Membership functions for different levels of the trust index Γ, job success rate Φ, and site defense capability Δ

Figure 5 shows the trust inference process using the membership functions in Fig.4. We consider initial values: $\Phi = 0.84$ and $\Delta = 0.26$, obtained from previous Grid application experiences. Two example *fuzzy inference rules* are given below for use in the inference process shown in Fig.5.

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

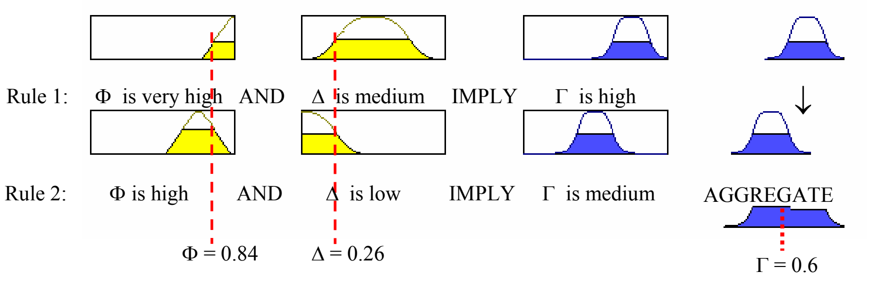

Fig. 5. Fuzzy logic inference between job success rate Φ and self-defense capability Δ to induce the trust index Γ of a resource site

All selected rules are inferred in parallel. Initially, the membership is determined by assessing all terms in the premise. The fuzzy operator 'AND' is applied to determine the support degree of the rules. The AND results are aggregated together. The final trust index $\Gamma = 0.6$ is generated by defuzzifying the aggregation. The "AGGREGATE" superimposes two curves to produce the membership function for Γ.

There are many other fuzzy inference rules that can be designed using various conditional combinations of the fuzzy variables, Φ and **Δ.**

4. Trust Integration Across Grid Resource Sites

We use trust integration across multiple Grid domains to model transitive trust relationship. Each site $\boldsymbol{S_j}$ maintains a *trust vector* $\boldsymbol{V_j = (t_{1j}, t_{2j},\ldots, t_{mj})^T}$ **.** The *trust index* $\boldsymbol{t_{ij}}$ for $1 \le \boldsymbol{i}, \boldsymbol{j} \le \boldsymbol{m}$ represents the trust of site $\boldsymbol{S_i}$ by site $\boldsymbol{S_j}$. $\boldsymbol{t_{ij}}$ is a fraction number with 0 representing the most risky case without any protection and 1 for a full trusted condition with the highest security assurance. Any value in between indicates a partially secured site. We define a *trust matrix* for ***m*** *resource sites* by an $\boldsymbol{m}\times\boldsymbol{m}$ square matrix $\boldsymbol{M = (V_1, V_2,\ldots, V_m) = (t_{ij})}$.

Trust update and trust propagation processes are specified in Algorithms 1 and Algorithm 2, respectively. We aim at reducing the site vulnerability and by upgrading its self-defense capability Δ. Suppose that the SeGO agent of site $\boldsymbol{S_i}$ has monitored all jobs executed on site $\boldsymbol{S_j}$ for some time to know its success rate and defense capability. Let $\boldsymbol{s_{ij}}$ be the new security stimulus between sites $\boldsymbol{S_i}$ and $\boldsymbol{S_j}$ at certain time instant. Equation (1) calculates the new trust index from the old value and present stimulus.

$$\boldsymbol{t_{ij}^{new}} = \alpha\, \boldsymbol{t_{ij}^{old}} + (1-\alpha)\, \boldsymbol{s_{ij}} \qquad \textbf{(1)}$$

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

```
Algorithm 1: Trust_Update(index_TTL reports, i, j)
(1)  $R_i$ calculate success rate of $R_j$:  $\Phi$ = number of success jobs/index_TTL;
(2)  $R_i$ assess defense rate $\Delta$ of $R_j$;
(3)  Calculate the stimulus value:  $S_{ij}$ = Fuzzy_inference($\Phi$, $\Delta$);
(4)  Calculate the new trust index:  $t_{ij}^{new} = \alpha t_{ij}^{old} + (1-\alpha)s_{ij}$;
(5)  if (($t_{ij}^{new} < t_{ij}^{old}$) or ($t_{ij}^{new}$ < average trust requirement))
         Enhance defense capability of $R_j$,  $\Delta(R_j) = \Delta(R_j) + \varepsilon(\Delta)$.

Algorithm 2: Trust_Propagation(i)
(1)  $R_i$ broadcasts $V_i$;
(2)  for $j$ = 1 to $i$-1, $i$+1 to $M$
(3)      $V_j^{new} = (m-1/m)V_j^{old} + V_i/m$ .
```

We introduce two simulation terms: the trust *index_TTL* and trust *vector_TTL*, to measure user applications submitted to each site. When site $\boldsymbol{S_j}$ accumulates *index_TTL* job reports from site $\boldsymbol{S_i}$, it updates the trust index $\boldsymbol{t_{ij}}$ using Eq.(1). With fuzzy trust quantification, the stimulus value $\boldsymbol{s_{ij}}$ is determined first. Then the new trust index $\boldsymbol{t_{ij}}$ is updated, accordingly. If the trust index decreases or it is lower than the average, the defense capability Δ of S_j is forced to increase by an amount ε characterized in Eq.(2).

$$\Delta(S_j) = \Delta(S_j) + \varepsilon(\Delta) \tag{2}$$

In Eq. (2), the increment ε(Δ) 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 trust integration is to enhance the trust index or security level at weak resource sites. Trust integration leads to normalization and equalization of trust indices at all sites. The trust vectors are broadcasted periodically. When a site S_j has accumulated *vector_TTL* of job execution reports, it broadcasts its trust vector to other sites. With m sites, the contribution from each site is roughly $1/m$. Algorithm 2 is used to calculate the new trust vector for site S_i by each resource site S_j for $j = 1,2,\ldots, m$.

5. Optimization of Trusted Resource Allocation

Based on the fuzzy trust model, we present below in Algorithm 3 the SeGO scheduler for optimized Grid resource allocation. The SeGO scheduler was developed under the following assumptions: (1) non-preemptive job scheduling, (2) divisible workload across Grid sites, and (3) space sharing in a batch mode over multiple jobs. Our SeGO scheduler is specified with a nonlinear programming model [5].

```
Algorithm 3: SeGO ($R_j$, $Job = (W, D, T, B)$)
Input:   Submit $Job = (W, D, T, B)$ to resource site $R_j$ at time $\tau$, $R_j$ requests
         resources from all $m$ sites.
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 to obtain available resources information 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
     $\sum_{i=1}^{m} x_i P_i L \ge W$ , $\sum_{i=1}^{m} x_i P_i L C_i \le B$ , and $0 \le x_i \le 1$;
(7)  for $i = 1$ to $m$    $W_i = x_i P_i L$;
(8)  return $(W_1, W_2, \ldots, W_m, L)$ with allocation $X = (x_1, x_2, \ldots, x_m)$.
```

A job is submitted with the descriptor $Job = (W, D, T, B)$, representing the *workload*, *execution deadline*, *minimum trust*, and *budget limit*. A job is required to complete execution before the posted deadline. Denote the current time instant by τ. This is the start time of a job execution. The *estimated job execution time* is denoted by $\boldsymbol{L} = \boldsymbol{D} - \tau$. Let $\boldsymbol{x_i}$ be the percentage of the peak power $\boldsymbol{P_i}$ allocated to the job $\boldsymbol{J}$. The product $\boldsymbol{x_i P_i}$ represents the actual power allocated. We define $\boldsymbol{W_i} = \boldsymbol{x_i P_i L}$ as the *workload* to be executed at site $\boldsymbol{R_i}$ for the job $\boldsymbol{J}$.

The input to Algorithm 3 is the successive job descriptions including the site and the time when job is submitted. After passing a qualification test, step (2)-(4), the unqualified sites are filtered out. The estimated execution time $\boldsymbol{L}$ is registered first. Then, the *resource allocation vector* $\boldsymbol{X} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m})$ is generated by optimizing the objective function or the *trusted performance/cost ratio* $\boldsymbol{E}$, defined below.

The numerator is the *aggregate computing power*, weighted by the trust index $\boldsymbol{t_{ij}}$ and allocated from $\boldsymbol{m}$ Grid sites. The denominator is the *total Grid service charge* for executing the job. The terms $\boldsymbol{P_i}$ and $\boldsymbol{C_i}$ are the computing power and service charge at site $\boldsymbol{R_i}$. The SeGO solution is obtained with a nonlinear programming solver, subject to the constraints listed in step 6.

$$\boldsymbol{E} = \sum_{i=1}^{m} W_i t_{ij} \Big/ \sum_{i=1}^{m} W_i C_i = \sum_{i=1}^{m} x_i P_i L t_{ij} \Big/ \sum_{i=1}^{m} x_i P_i L C_i \qquad (3)$$

Algorithm 4 specifies the trust integration process, in which $\boldsymbol{n}$ jobs are mapped to $\boldsymbol{m}$ sites. The trust vectors are propagated and integrated periodically. If a job is submitted to $\boldsymbol{R_j}$, this site is responsible to dispatch workload to all sites and monitors the job execution. Once a job is finished, the occupied resources are released for other jobs. User applications can resubmit their jobs, if the earlier execution was unsuccessful. The trust integration process includes trust update (Algorithm 1), trust propagation (Algorithm 2), and SeGO optimization (Algorithm 3). The inputs to this algorithm are jobs submitted at all sites. The output is the trusted resource allocation and the updated trust vectors.

```
Algorithm 4: Trust integration for optimized resource allocation
Input:  $n$ jobs submitted at $m$ resource sites.
Output: Resource allocation for jobs and updated trust vectors for all sites.
(1)  Do until (all submitted jobs are executed)
(2)    if ($\tau$ = arrival time of current $Job = (W, D, T, B)$)
(3)      $Job$ is put in the job queue of $R_j$;
(4)      $(W_1, W_2, \ldots, W_m, L) \leftarrow SeGO(R_j, Job)$;
(5)      for $i = 1$ to $m$   resource reservation, i.e., $P_i = P_i - W_i/L$;
(6)    end if
(7)    if ($R_j$ gets the previous $Job = (W, D, T, B)$ report at time $\tau$)
(8)      for $i = 1$ to $m$
(9)        resource release, i.e., $P_i = P_i + W_i/L$;
(10)       if ($R_j$ accumulates index_TTL job reports from $R_i$)
(11)         Trust_Update (index_TTL reports, $i, j$);
(12)       if ($R_j$ accumulates execution reports for vector_TTL jobs)
(13)         Trust_Propagation ($j$);
(14)     end for
(15)   end if
(16) end do
```

6. Simulation Results on Trusted Grid Resource Allocation

We have developed a discrete-event simulator at USC to simulate the trust integration and resource optimization processes. We simulated $\boldsymbol{n} = 300$ jobs running on $\boldsymbol{m} = 6$ Grid resource sites. Each resource site is configured with computing power, which is set between 1 Tflop/s and 5 Tflop/s, randomly. Each site is configured with a site reliability and intrusion defense capability in the range (0, 1).

Jobs are mapped evenly across sites, and all job arrivals are modeled by a Poisson distribution with an inter-arrival time of 10 minutes. The job workload demand varies between 4 Tflop to 50 Tflop. The deadline varies between 4 minutes and 20 minutes after the job is submitted. The minimum trust is set in the range (0.4, 0.7) randomly. Both the resource unit service charge and user application budget limitations are set between $180K/Tflop and $320K/Tflop, randomly.

Figure 6 depicts the variation of the trust index values at 6 resource sites, $\boldsymbol{R}_1$ through $\boldsymbol{R}_6$. The intial trust index values at step 0 vary from 0.07 to 0.77 for sites $\boldsymbol{R}_1$ through $\boldsymbol{R}_6$ along the Y-axis. The x-axis represents the trust integration step taken during simulation runs. The average trust index at each site increases steadily after each step. Through the process, all trust indices grow to the range (0.7, 0.93) at step 5.

In the best case, the lowest index value of 0.07 at site $\boldsymbol{R}_1$ increases to 0.7 in 5 steps. This corresponds to a security enhancement of 90% = (0.7 - 0.07)/0.7 for site $\boldsymbol{R}_1$. In the worst case for site $\boldsymbol{R}_6$, the trust index is upgraded from 0.7 to 0.93 in 5 steps. There is a normalization effect of the trust integration process, which brings the security levels of all sites to almost the same high level.

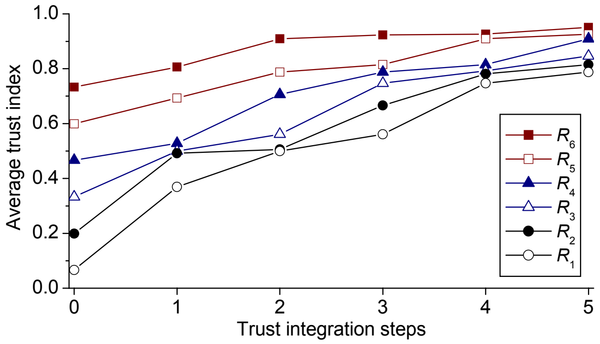

Fig. 6. Variation of the trust indices of six resource sites after five trust integration steps

We present in Fig.7 and Fig.8 two scatter plots of the *performance/cost ratio*. The two scatter plots result from running the SeGO simulation under different trust management polices. Each triangle represents the performance/cost ratio of one job. Both figures plot the Grid performance under limited budget with initial trust values ranging from 0.07 to 0.77 given at step 0 in Fig.6.

Figure 7 depicts performance/cost ratio E of 300 jobs with fixed trust, meaning no security upgrade over the resource sites. Figure 8 plots E with trust integration to upgrade the defense capabilities at six resource sites. We observed two job groups in these plots. One group consists of those dropped jobs due to short of resources before the deadline expired. Those jobs are represented with $E = 0$ along the X-axis. The second job group contains the successful executed jobs. There are 76 dropped jobs in Fig.7 and 18 dropped jobs in Fig.8 out of 300 jobs simulated. This translates to a *job drop rate* of 76/300 = 25.3% in Fig.7 and 18/300 = 6% in Fig.8.

In Fig.7, the E-plot for successful jobs varies from 1.67 to 2.71 Tflop/$1M with an average $E = 2.27$ Tflop/$1M. In Fig.8, the successful jobs achieve $E = 1.67$ to 3.57 Tflop/$1M with an average $E = 2.92$ Tflop/$1M. Overall, the scatter plot in Fig.7 shows almost no increasing trend as more jobs are submitted. However, the E-plot in Fig.8 increases steadily as more jobs are submitted. Considering the last 50 jobs, we achieved $E = 2.94$ to 3.57 Tfop/$1M in Fig.8. We observe an *improvement factor* by 114% = (3.57-1.67)/1.67 for the best-case scenario.

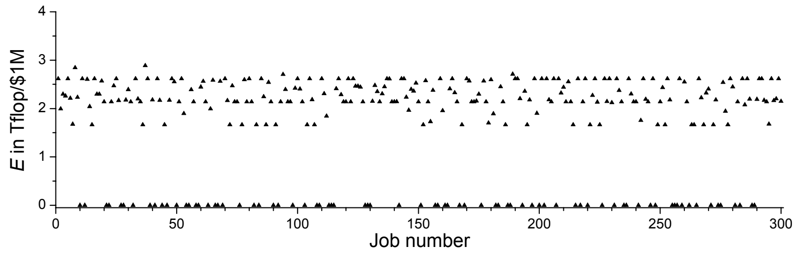

Fig. 7. Grid performance/cost ratio for 300 jobs allocated to six resource sites with fixed trust index and no site security reinforcement

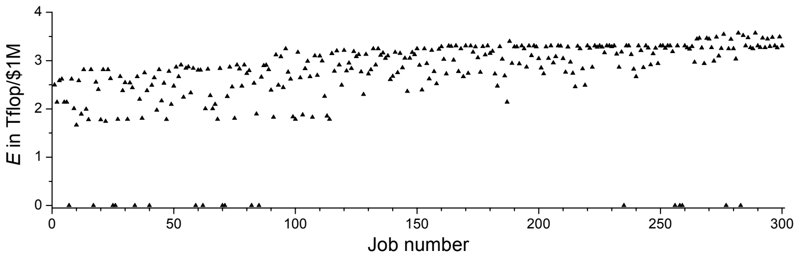

Fig. 8. Improved Grid performance/cost ratio for 300 jobs allocated to 6 resource sites after trust integration and security upgrade

In summary, our trusted resource allocation (Fig.8) shows a 76% - 114% improvement in *Grid performance/cost ratio E*. The *job drop rate* is reduced by (76-18)/76 = 75% in favor of trust integration solution. On the average E, a performance gain of 28% = (2.92-2.27)/2.27 was resulted from trusted resource allocation. As a matter of fact, the trust-integration process is at work very early on. After the submission of the first 15 jobs, the E starts to climb, and achieves more than 3.0 Tflop/$1M at the 100^{th} job. The results clearly demonstrate the effectiveness of trust integration. Trusted Grid sites accommodated 94% = 1 – 6% of 300 user jobs.

Utilization rate is defined as percentage of allocated resources among all available resources. The utilization rate for resources with fixed trust values remains at the constant level at 40% during the simulation runs. The utilization rate for resources with integrated trust values varies from a low of 48.1% to a high of 92.6%. The utilization of Grid resources increases with more jobs submitted. These results demonstrate significant gain in Grid performance through optimized resource allocation and aggressive security reinforcement by trust integration.

Table 1. Utilization of Grid Resources at Six Sites for the Execution of 300 Jobs

Grid resource utilization rate	**Job Number**					
	1 - 50	51 - 100	101 - 150	151 - 200	201 - 250	251 - 300
With fixed trust	39.4%	45.1%	43.0%	34.9%	45.1%	38.4%
With trust integration	48.1%	78.0%	65.4%	84.2%	92.6%	82.9%

7. Conclusions and Suggestions for Further Research

This work offers the first step towards trusted Grid computing. In several recent reports from USC Internet and Grid Computing Laboratory, one can find comprehensive treatment of the GridSec architecture [9], Internet traffic datamining for automated intrusion detection [13], and trusted Grid resource allocation [14]. We summarize below research findings and make a few suggestions for further research.

- Fuzzy trust integration reduces platform vulnerability and guides the defense deployment across Grid sites. Our VPN-supported trust integration is meant to enforce security in Grids beyond the use of PKI services [2, 9, 14]. Comprehensive simulation results were reported in [14] to prove the effectiveness of the SeGO scheduler for trusted resource allocation in computational Grids.
- Self-defense toolkits are needed to secure Grid computing [9]. We have suggested the use of distributed firewalls, packet filters, virtual private networks, and intrusion detection systems at Grid sites. A new anomaly-based, intrusion detection system was developed with datamining of frequent traffic episodes in TCP, UDP, and ICMP connections as reported in [13].
- Regarding future research directions, we suggest to integrate the SeGO scheduler with other Grid job/resource management toolkits such the Globus/GRAM, AppLex, and NimRod/G [2, 4]. Grid security policies and Grid operating systems are needed to establish truly secure Grid computing environment [15].

References

1. F. Azzedin and M. Maheswaran, “Towards Trust-Aware Resource Management in Grid Computing Systems”, *Proc. of Int’l* Symp. *on Cluster Computing and the Grid*, 2002.
2. F. Berman, G. Fox, and T. Hey, (Editors), *Grid Computing: Making The Global Infrastructure a Reality*, John Wiley & Sons, 2003.

3. Butt, S. Adabala, N. Kapadia, R. Figueiredo, and J. Fortes, “Fine-Grain Access Control for Securing Shared Resources in Computational Grids”, *Proceedings of Int’l Parallel and Distributed Processing Symposium*, April 2002.
4. R. Buyya, M. Murshed and D. Abramson. “A Deadline and Budge Constrained Cost-Time Optimization Algorithm for Scheduling Task Farming Applications on Global Grids”, *Int’l Conf. on Parallel and Distributed Processing Techniques and Applications*, 2002.
5. R. Byrd, M. E. Hribar, and J. Nocedal. “An Interior Point Method for Large Scale Nonlinear Programming”, *SIAM Journal on Optimization*, Vol. 9, 1999, pp. 877-900.
6. K. Czajkowski, I. Foster, and C. Kesselman, “Resource Co-Allocation in Computational Grids”, *Proc. of the 8th IEEE Int’l Symp. on High Perf. Distributed Computing*, 1999.
7. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, “A Security Architecture for Computational Grids”, *Proc. of ACM Conf. on Computer and Comm. Security*, 1997.
8. M. Humphrey and M. Thompson, “Security Implications of Typical Grid Computing Usage Scenarios”, *Proceedings of Int’l Symposium on High Performance Distributed Computing* (HPDC), San Francisco, CA. Aug. 7-9, 2001.
9. Hwang, et al, “The GridSec and Netshield Architecture for Securing Grid and Distributed Computing”, *Technical Report 2004-15*, USC Internet and Grid Computing Lab, 2004.
10. B. Kosko, Fuzzy Engineering, Prentice Hall, 1997.
11. H. Liu and J. Shen, “A Mission-Aware Trust Model for Grid Computing Systems”, *Int’l Workshop on Grid and Cooperative Computing,* Sanya, China, Dec. 26, 2002.
12. Manchala, “E-Commerce Trust Metrics and Models*”, IEEE Internet Computing*, March 2000, pp. 36-44.
13. M. Qin and K. Hwang, “Anomaly Intrusion Detection by Internet Datamining of Traffic Episodes”, *Technical Report No. 2004-6,* USC Internet and Grid Computing Lab, also submitted to *ACM Trans. on Information and System Security* (TISSec), March 2004.
14. S. Song and K. Hwang, “ Security Binding for Trusted Resource Allocation in Computational Grids”, *Technical Report 2004-8*, USC Internet and Grid Computing Lab, also submitted to *IEEE Trans. on Parallel and Distributed Systems,* May 2004.
15. Z. Xu, W. Li, H. Fu, and Z. Zeng,“ The Vega Grid Project in China“, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, April 2003. http://link.springer.de/link/service/series/0558/papers/2436/24360228.pdf

Biographical Sketches

Shanshan Song received her BS degree in Computer Science from a special class for gifted young in the University of Science and Technology of China, Hefei, in 2001, and the MS degree in Computer Science from University of Southern California (USC) in 2003. Currently, She is pursuing the Ph.D. degree in Department of Computer Science at USC. Her research interest lies primarily in the area of network security and dynamic resource allocation for computational Grids. She can be reached at **shanshas@usc.edu.**

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, cluster and Grid computing systems. Presently, he leads a NSF-supported ITR Grid security project at USC. The GridSec group develops security-binding techniques for trusted outsourcing in Grid computing. They build self-defense software toolkits for protecting Grid and distributed computing resources. Dr. Hwang can be reached at **kaihwang@usc.edu** or through the URL: **http://GridSec.usc.edu/Hwang.html**

Mikin Macwan received the B.E. degree from Pune University, India in 1999, and the M.S. degree in Computer Science from Texas A&M University - College Station, in 2001. After working as a Software Engineer at Sun Microsystems, CA (2001 - 2002), he joined the Computer Science Program at USC. He is now pursuing the Ph.D. degree at USC. His primary areas of research are Network Security and Intrusion Detection and Response Systems. He can be reached at **macwan@usc.edu.**