IEICE TRANS. INF. & SYST., VOL.E94–D, NO.1 JANUARY 2011

PAPER

Trust Management of Grid System Embedded with Resource Management System

Sherihan ABU ELENIN[†a)], ***Nonmember and*** **Masato KITAKAMI**[†b)], ***Member***

SUMMARY Recently, Trust has been recognized as an important factor for Grid computing security. In this paper, we propose a trust model in Grid system. It consists of Application Domain (AD), Client Domain (CD), Resource Domain (RD), and Trust Manager (TM). TM controls the relationship between RD and CD depending on the trust level value of each client and classification of each resource. Performance criteria are makespan and utilization. We evaluated our trust model in six scheduling algorithms in nine scenarios. The simulation results show that the proposed trust model improves the performance in all scheduling algorithms.

key words: *Grid computing, security, trust model, resource management system*

1. Introduction

The security problem is a hot topic in Grid research due to the dynamics and uncertainty of Grid system. There are three entities defined as users, applications and resources in Grid environment. In such situation, users are vulnerable to risk because of potential incomplete or distorted information provided by malicious resources, and as Grid system grows tremendously in size, the possibility of users attacking the network by providing aggressive or vicious applications will increase greatly. Trust management is an effective method to maintain the credibility of the system and keep honesty of entities [8].

Trust [1]-[3] is the firm belief in the competence of an entity to act as expected such that this firm belief is not a fixed value associated with the entity but rather it is subject to the entity's behavior and applies only within a specific context at a given time. Trust management (TM) is collecting, codifying, analyzing, and evaluating evidence relating to competence, honesty, security, or dependability with the purpose of making assessments and decisions regarding trust relationships [9]. Trust management systems (TMS) must support analysis of trust and recommendation specifications to detect conflicts and inconsistencies and support trust queries related to decision making.

In this paper, we focus on trust management in Grid computing. The proposed model is based on the trust model proposed by Azzedin and Maheswaran [1]–[5]. They measured the performance of Grid system by applying their trust model in a resource management system. They worked with Minimum Completion Time heuristic, Min-min heuristic, and Sufferage heuristic algorithms. In this paper, we examined six scheduling algorithms with our trust model. The new point in the proposed trust model is the computing and evaluating trust. Trust manager's operations in the proposed trust model make more control and management in the system than Trust agent's operations in the conventional trust model.

Manuscript received November 16, 2009.
Manuscript revised July 06, 2010.

[†]The authors are with the Graduate School of Advanced Integration Science, Chiba University, Chiba-shi, 263–8522 Japan.
a) E-mail: sherihan@graduate.chiba-u.jp
b) E-mail: kitakami@faculty.chiba-u.jp

DOI: 10.1587/transinf.E94.D.42

2. Related Work

There is a lot of research on the trust in distributed systems. Here we just mention some works that are deeply related to our paper.

Abdul-Rahman and Hailes [10] proposed a trust model for computing the trust for an agent in a specific context based on the experience and recommendations. They applied and implemented their trust model in P2P networks. Trust can have only four possible values; very trustworthy, trustworthy, untrustworthy, and very untrustworthy. Each agent stores the trust values for the agents with him/her interacts and the recommender trust with respect to another agent. So each agent has to store all history of past experiences and received recommendations.

Azzedin and Maheswaran [1]–[5] defined the notion of trust as consisting of identity trust and behavior trust. They separate the "Grid domain" into a "Client domain" and a "Resource domain". They view trust in two steps: verifying the identity of an entity and what that identity is authorized to do, and monitoring and managing the behavior of the entity and building a trust level based on that behavior. The way they calculate trust is limited in terms of computational scalability, because they try to consider all domains in the network.

C. Lin, V. Varadharajan, Y. Wang, and V. Pruthi [11] presented trust management architecture for trust enhanced Grid security. The trust model is capable of capturing various types of trust relationships that exist in a Grid system and providing mechanisms for trust evaluation, recommendations and update for trust decisions. The outcomes of the trust decisions can then be employed by the Grid security system to formulate trust enhanced security solutions.

3. Conventional Trust Model

Trust evaluation has always been a challenge for online communities [10]. Most of the research focuses on P2P and

Internet. The problems of managing trust in Grid environments are discussed by Azzedin and Maheswaran [1]–[5]. Figure 1 shows the overall trust model in which the Grid is divided into Grid domains (GDs). They associate two virtual domains with each GD, namely a resource domain (RD) to signify the resources within the GD and a client domain (CD) to signify the clients within the GD. As RDs and CDs are virtual domains mapped onto GDs, some instances of RDs and CDs can map onto the same GD. Trust agents exist in each GD with mechanisms to update the GDs' trust tables, allow entities to join GDs and inherit their trust attributes, and apply a decay function to reflect the decay of trust between domains.

They define the trust level Table (TLT) as it is built on past experiences and is given for a specific context. TLT as shown in Table 1 has six values from very low trust level to extremely high trust level. From TLT, they can compute the offered trust level (OTL) for the composite activity between X and Y. There are two required trust levels (RTLs), one from the client side and the other from the resource side. If the OTL is greater than or equal to the maximum of client and resource RTLs, then the activity can be proceed with no additional overhead. Otherwise, there will be additional security overhead involved in supplementing the OTL to meet the requirements. The trust level values used in Table 1 range from very low trust level denoted as A, to extremely high trust level denoted as F. They can compute the expected trust supplement (ETS) for different RTL and OTL values. The ETS values are given by RTL − OTL. The ETS value is zero, when RTL − OTL < 0.

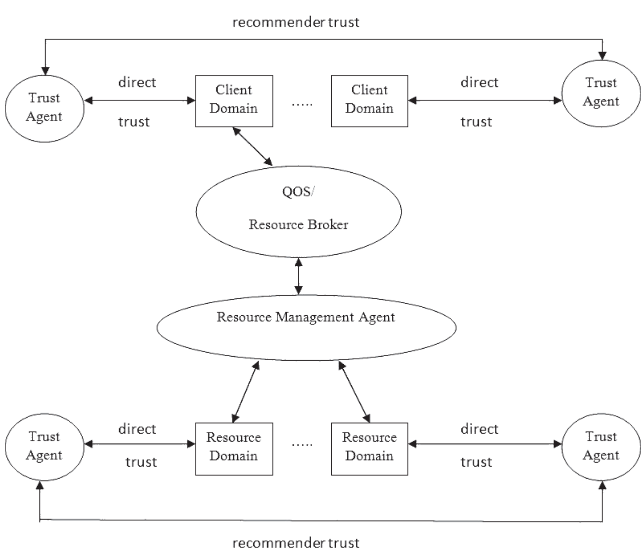

Fig. 1 Components of conventional trust model.

Table 1 Description of trust level table.

Trst Level (TL)	Description
A	very low trust level
B	low trust level
C	medium trust level
D	high trust level
E	very high trust level
F	extremely high trust level

4. Proposed Trust Model

4.1 Overview

Figure 2 shows the proposed trust model in which the Grid is divided into Grid domains (GDs). GD consists of application domain (AD), resource domain (RD), and client domain (CD). The functions of CD and RD are the same of the trust model in [1]–[5]. Every client has a trust level value. This value is one point real value from 0 to 1 to measure the trust value for every client; those values are different from the model in [1]–[5]. The examples of trust level values are 0.3, 0.4, 0.9, etc. These values are changed depending on the trustee of the client. If he is trusted, his trust level value will be incremented by 0.1 until it reaches the maximum trust value; 1. If he isn't trusted as shown in Fig. 3, his trust level value will be decrement by 0.1 until it reaches 0. This means this client isn't trusted and can't use the system at all. AD is added to execute any resources. The system always has the direct relationship between CD and RD. We have some services as examples in RD such as print file, open file for readable, copy file, rename file, move file, delete file, and open file for writable. The system also has direct relationship between CD and AD, but not examined yet.

Trust Manager (TM) is replaced in the model. TM's operations consist of Trust Locating, Trust Computing, and Trust Updating. Trust Locating consists of Authentication Controller and Certificate Authority Controller. Trust Locating is responsible of authentication of clients and checks the certificate authority for every client. Trust computing

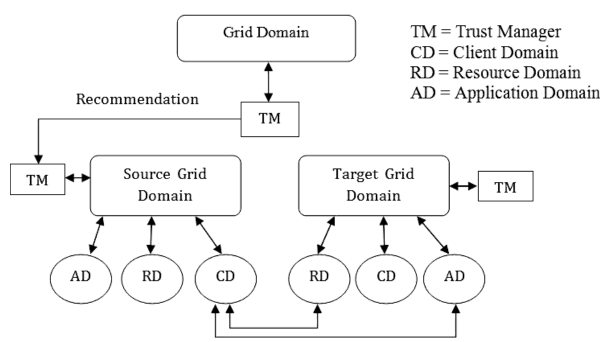

Fig. 2 Proposed trust model.

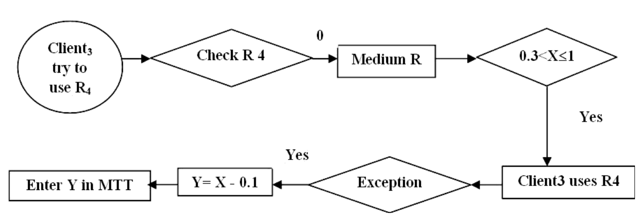

Fig. 3 Example of Client3 trying to use R4.

Table 2 Comparison between conventional and proposed trust model.

Conventional Trust Model	Proposed Trust Model
GD is divided into RD and CD.	GD is divided into RD, CD, and AD.
Trust agent's functions are updating the GDs' trust tables, allowing entities to join GDs, and applying a decay function.	Trust manager's operations consist of Trust Locating, Trust Computing, and Trust Updating. Every operation consists of sub-operations.
Trust level's values are from 1 to 6 (A-F); from very low trust level to extremely high trust level. These values are not changed at all.	Trust level's values are from 0 (untrusted) to 1 (trusted). These values are changed to reflect the trust level of every client in every context.
There is no classification of resources.	There is classification of resources to access them easily; low, medium, or high.

consists of Authorizing Controller, Trust Relationship Controller, and Environment Evaluation. Trust Computing determines the allowed resources for every client depending on his/her trust level value and computes components required for the evaluation of trust relationships. Trust Updating returns to trust manager all updates in the system.

The proposed trust model likes the trust model [1]–[5] in the structure of the Grid but they are completely different in the management of the system and the way the trust is computed as in Table 2.

4.2 Trust Manager

The following is the procedure of TM works:

1. Trust Manager is the first component to be activated and it will assign a task to Trust Locating.
2. Trust Locating assigns a task to Authentication Controller, and Authentication Controller wants to know "who are you?"
3. Authentication Controller returns the authentication of the client to Trust Locating.
4. Trust Locating requires certificate authority for both host and client and sends the request to Certificate Authority Controller.
5. Certificate Authority Controller returns the Certificate Authority to Trust Locating.
6. Trust Locating returns information of locating of trust to Trust Manager.
7. Trust Manager requires trust computing to compute and evaluate the trust relationship and it will assign a task to Trust Computing.
8. Trust Computing assigns a task to Authorizing Controller to know the allowed resources to specific client and checks the resource classification which is classified into three levels: low, medium or high resource. If it is low, medium, or high, it will have value -1, 0, 1 respectively. In Table 3 there are some of these resources classification.
9. Authorizing Controller returns back the allowed resources to Trust Computing.

Table 3 Examples of resources classification.

R_{number}	Resource	Fuzzy Logic Value
R1	Print	-1
R2	Open as readable(R)	-1
R3	Copy	-1
R4	Rename	0
R5	Move	0
R6	Delete	1
R7	Open as writable(W)	1

Table 4 Proposed Trust Level Table (TLT).

Client	Trust Value (X)
Client1	0.3
Client 2	1
Client 3	0.5
Client 4	0.3
Client 5	0.2

Table 5 Modification Trust Table (MTT).

Client	Trust Value (X)
Client1	0.3
Client 2	1
Client 3	0.4
Client 4	0.4
Client 5	0.2

10. Trust Computing sends a request to Trust Relationship Controller to get the trust value (X) from Trust level Table (TLT) as in Table 4 to specific client to know if he/she can use the required resource. If the client's trust value less than or equal 0.3, he/she can use the low resources. If the client's trust value greater than 0.3 and less than or equal 0.8, he/she can use the medium and low resources. If the client's trust value greater than 0.8 and less than or equal 1, he/she can use the high, medium and low resources and can also execute applications.
11. Trust Relationship Controller returns the trust value of client and he/she can use the required resource or not to Trust Computing.
12. Trust Computing requires Environment Evaluation to evaluate the trust. After evaluation, computing the new trust value (Y); computation depends on if there is exception or not. If there is exception the trust value will be decreased until it becomes 0, it means this client becomes untrusted, but if there is no exception the trust value will be increased. If the trust value is the maximum trust value (X=1) it will not increase; and enter Y in Modification Trust Table (MTT) as in Table 5.
13. Environment Evaluation returns the new trust value (Y) to Trust Computing.
14. Trust Computing integrates the results of 9, 11, and 13 and returns computing result to Trust Manager.
15. Trust Manager sends two trust values; X from TLT and Y from MTT; to Trust Updating.
16. Trust Updating makes the new updates and computes the trust bit value depending on the increment or decre-

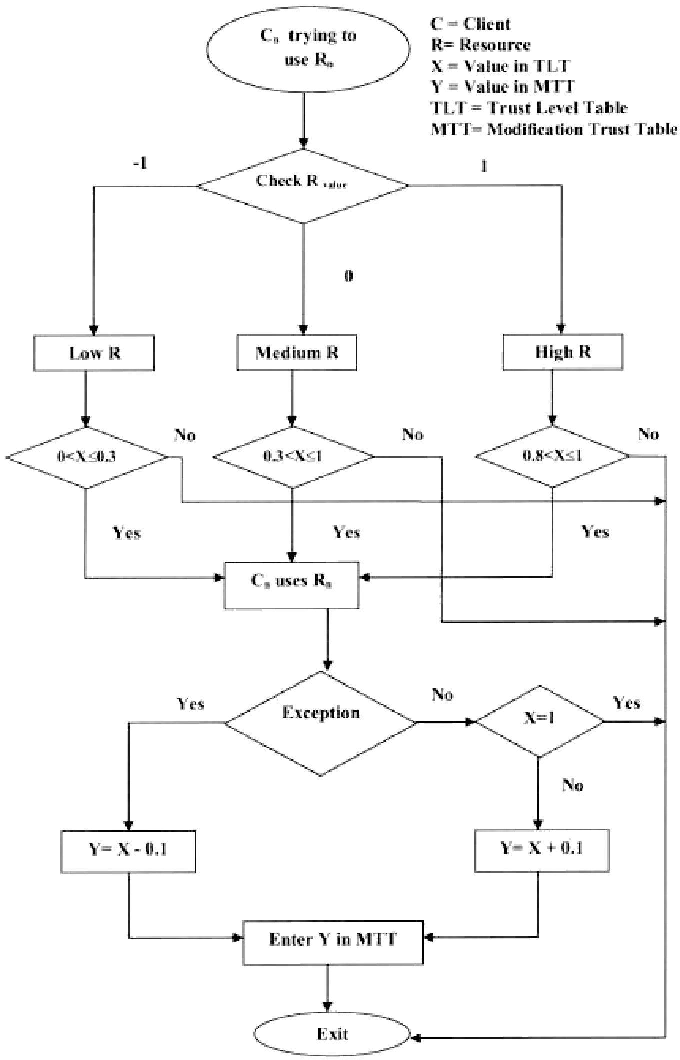

Fig. 4 Flow chart of computing trust.

Table 6 Trust Bit Table (TBT).

	R1	R2	R3	R4	R5	R6	R7
C1	1	1	0				
C2	1	1	1	1	1	1	1
C3	1	1	1	0	1		
C4	1	0	1				
C5	1	1	1				

ment in the trust value. If there is increment, the trust bit will be 1, but if there is decrement, the trust bit will be 0. The new value of trust bit is entered in Trust Bit Table (TBT) as in Table 6 and is returned back to Trust Manager.

The heart of the proposed trust model is Trust Manger. TM makes its work in three stages and every stage has substages to do. The new thing in the model is how to compute the trust depend on experience. Experience can depend on not only user history but also certificate of the others; lookup in the Table 6. Figure 4 shows the flow chart of computing trust, starting with authorization, trust relationship, and ending with evaluation.

4.3 System Environment

Our Grid platform consists of: 1) Hardware Components: Nodes: 5 PCs (Intel Pentium4 2.2 GHz processor, Intel RAM 256 MB) and 10 PCs (Intel Atom 1.66 GHz processor, Intel RAM 2 GB), and Interconnection Network: Fast Ethernet. 2) Grid Middleware: Globus Toolkit 4.2.1. 3) Software Components: Operating System: Linux Fedora 10, and Tools: Programs written in Java, packages from ponder policy language, and Apache Ant for Java-based build tool.

The system is divided into two domains (D1 and D2). D1 consists of G1, G2, G3, G4, G5, G6, G7, and G8. D2 consists of G1, G2, G3, G4, G5, G6, and G7. Trust Manger of D1 is existed in G1 and that of D2 in G1. Nodes G5 and G7 are existed in the two domains. Every domain has resource domain, client domain, and application domain with of course TM. In the system, always every node is called with its domain name such as G5: D2.

5. Resource Management Systems

Resource Management Systems (RMSs) are used to govern the execution of the tasks that arrive for service [6]. Grid task scheduling is one of the most important parts in Grid resource management system. The performance of algorithm is the key performance goal to evaluating Grid. The process of matching and scheduling tasks is referred to as mapping. There are two types of mapping heuristics: Immediate mode heuristics, and Batch mode heuristics.

5.1 Batch Mode Heuristics

Tasks are collected into a set that is examined for mapping at prescheduled times called mapping events [6]. The independent set of tasks that is considered for mapping at the mapping events is called a meta-task. A meta-task can include newly arrived tasks and the ones that were mapped in earlier mapping events but did not begin execution. It considers a task for mapping at each mapping event until the task begins execution. We discuss here in this paper 3 types of batch mode heuristics.

Min-min heuristic algorithm: This heuristic begins with the set U of all unmapped tasks. Then the set of minimum completion times, M, is found. Next, the task with the overall minimum completion time from M is selected and assigned to the corresponding machine and the workload of the selected machine will be updated. And finally the newly mapped task is removed from U and process repeats until all tasks are mapped [7].

Max-min heuristic algorithm: It is very similar to min-min, but chooses the maximum expected execution time.

Sufferage heuristic algorithm: In this heuristic for each task, the minimum and second minimum completion time are found in the first step. The difference between these two values is defined as the sufferage value. In the second step, the task with the maximum sufferage value is assigned to the corresponding machine with minimum completion time [7].

5.2 Immediate Mode Heuristics

A task is mapped onto a machine as soon as it arrives at the mapper [6]. Each task is considered only once for matching and scheduling, i.e. the mapping is not changed once it is computed. It considers a task for mapping only once. We discuss here in this paper 3 types of immediate mode heuristics.

Opportunistic Load Balancing heuristic algorithm (OLB): This method assigns a job to the earliest idle machine without taking into account the execution time of the job in the machine. If two or more machines are available at the same time, one of them is arbitrarily chosen [6].

Minimum Completion Time heuristic algorithm (MCT): This method assigns a job to the machine yielding the earliest completion time. When a job arrives in the system, all available resources are examined to determine the resource that yields the smallest completion time for the job [6].

Minimum Execution Time heuristic algorithm (MET): This method assigns a job to the machine having the smallest execution time for that job. Unlike MCT method, MET does not take into account the ready times of machines [6].

6. Results and Discussions

6.1 Performance Evaluation

The performance of the proposed trust model which based on six types of scheduling algorithm was evaluated. Two metrics were used to evaluate the performance of the model. The first one is the makespan; the makespan is defined as $C_{max} = \max_j C_j$, where C_j is the completion time. The second criterion is utilization of machines for all the results discussed here. The experiments are performed on a number of meta-tasks of size 50, 100, and 1000. We have nine scenarios:

- Scenario I: 50 Tasks, 5 Machines, Consistent, Low Task, Low Machine Heterogeneity (LoLo);
- Scenario II: 50 Tasks, 10 Machines, Consistent, Low Task, Low Machine Heterogeneity (LoLo);
- Scenario III: 50 Tasks, 15 Machines, Consistent, Low Task, Low Machine Heterogeneity (LoLo);
- Scenario IV: 100 Tasks, 5 Machines, Consistent, Low Task, Low Machine Heterogeneity (LoLo);
- Scenario V: 100 Tasks, 10 Machines, Consistent, Low Task, Low Machine Heterogeneity (LoLo);
- Scenario VI: 100 Tasks, 15 Machines, Consistent, Low Task, Low Machine Heterogeneity (LoLo);
- Scenario VII: 1000 Tasks, 5 Machines, Consistent, Low Task, Low Machine Heterogeneity (LoLo);
- Scenario VIII: 1000 Tasks, 10 Machines, Consistent, Low Task, Low Machine Heterogeneity (LoLo);
- Scenario IX: 1000 Tasks, 15 Machines, Consistent, Low Task, Low Machine Heterogeneity (LoLo).

In the first part of the experiment, we computed the makespan of meta-tasks for nine scenarios. We try to reduce the makespan to make improvement by the proposed trust model. All algorithms were tested twice. Once when the Grid system didn't apply the proposed trust model, and another when it has proposed trust model. We computed the improvement of the proposed trust model in all algorithms.

In the second part of the experiment, we computed the machine utilization. Utilization is the ratio of time a system is busy divided by the time it is available. Utilization is a useful measure in evaluating performance.

We note that the proposed trust model is successes with Batch mode heuristics than Immediate mode heuristics. The reason is the Batch mode environment is dynamic in mapping meta-tasks, and this is suitable for the dynamic change in the trust values in the proposed trust model. But the static mapping of meta-tasks in Immediate mode environment isn't suitable with changing trust values in the pro-

Table 7 Performance evaluation of Scenario I (50Tasks × 5Machines).

Algorithm	Using Trust	Makespan (sec)	Utilization (%)	Improvement (%)
Min-min	No	3268	93.19	62
	Yes	1241	92.67	
Max-min	No	3979	99.31	35
	Yes	2567	99.16	
Sufferage	No	3563	94.30	56
	Yes	1541	95.47	
OLB	No	7368	92.30	7
	Yes	6857	92.31	
MET	No	21220	74.33	3
	Yes	20595	74.33	
MCT	No	4244	93.91	45
	Yes	2319	93.99	

Table 8 Performance evaluation of Scenario II (50Tasks × 10Machines).

Algorithm	Using Trust	Makespan (sec)	Utilization (%)	Improvement (%)
Min-min	No	2363	90.31	65
	Yes	810	89.33	
Max-min	No	2857	96.47	47
	Yes	1495	96.22	
Sufferage	No	2541	91.20	60
	Yes	999	91.71	
OLB	No	5491	87.90	10
	Yes	4911	87.90	
MET	No	15444	69.91	8
	Yes	14088	69.88	
MCT	No	3157	89.90	49
	Yes	1595	90.01	

Table 9 Performance evaluation of Scenario III (50Tasks × 15Machines).

Algorithm	Using Trust	Makespan (sec)	Utilization (%)	Improvement (%)
Min-min	No	1534	82.40	72
	Yes	420	82.19	
Max-min	No	1789	89.99	50
	Yes	883	89.34	
Sufferage	No	1581	84.11	69
	Yes	490	84.34	
OLB	No	3084	81.11	11
	Yes	2728	81.18	
MET	No	9610	60.48	10
	Yes	8597	60.30	
MCT	No	1922	81.99	50
	Yes	961	82.12	

posed trust model.

When number of meta-tasks is 50 as shown in Tables 7-9 (Scenario I, Scenario II, and Scenario III), Min-min algorithm is the best algorithm in reducing makespan. The improvement of it is increased with increasing the number of machines. The highest improvement value is recorded in Scenario III with Min-min algorithm (72%). The small number of tasks and the large number of machines are the reasons in this result. Beside these reasons, it selects minimum completion time tasks, so it helps in reducing makespan. MET algorithm is the worst algorithm in reducing makespan as shown in Tables 7-9. If number of machines is 15, the improvement of makespan is 10% and this is the smallest value according to the number of tasks; 50, and number of machines; 15. We think this result is because the MET algorithm doesn't take into account the ready times of machines.

Table 10 Performance evaluation of Scenario IV (100Tasks × 5Machines).

Algorithm	Using Trust	Makespan (sec)	Utilization (%)	Improvement (%)
Min-min	No	6372	96.15	31
	Yes	4394	95.93	
Max-min	No	8018	99.82	14
	Yes	6864	99.64	
Sufferage	No	6594	97.21	30
	Yes	4598	97.45	
OLB	No	15593	94.62	4
	Yes	14966	94.63	
MET	No	35085	76.46	1
	Yes	34670	76.46	
MCT	No	7017	96.95	30
	Yes	4934	96.99	

Table 11 Performance evaluation of Scenario V (100Tasks × 10Machines).

Algorithm	Using Trust	Makespan (sec)	Utilization (%)	Improvement (%)
Min-min	No	4594	91.45	36
	Yes	2896	91.11	
Max-min	No	5770	96.81	18
	Yes	4000	96.62	
Sufferage	No	4585	94.91	35
	Yes	2970	94.99	
OLB	No	10717	89.92	6
	Yes	9999	89.94	
MET	No	25593	70.11	2
	Yes	24999	70.12	
MCT	No	4994	91.61	33
	Yes	3311	91.91	

In the second part of the experiment as shown in Tables 7-9, Max-min algorithm is the best algorithm in introducing good utilization; 99.16%, 96.22%, and 89.34% in Scenario I, Scenario II, and Scenario III respectively. It selects the maximum completion time tasks, so the waiting time is always small to the remaining tasks and it also makes the machines always busy. MET algorithm is the worst algorithm in introducing good utilization. The reasons are discussed above. We note that when number of machines is increased, the utilization is decreased. Utilization is increased by increasing number of tasks and decreasing number of machines.

When number of meta-tasks is 100 as shown in Tables 10-12 (Scenario IV, Scenario V, and Scenario VI), Min-min algorithm and sufferage algorithm are the best algorithms in reducing makespan. MET algorithm is the worst algorithm in reducing makespan as shown in tables 10-12. We note that all improvements of all algorithms are increased when number of machines is increased. For example, Sufferage algorithm is improved by 30%, 35%, and 37% when number of machines is 5, 10, and 15 respectively.

In the second part of the experiment as shown in Ta-

Table 12 Performance evaluation of Scenario VI (100Tasks × 15Machines).

Algorithm	Using Trust	Makespan (sec)	Utilization (%)	Improvement (%)
Min-min	No	2686	84.54	40
	Yes	1597	84.31	
Max-min	No	3709	91.88	20
	Yes	2932	91.76	
Sufferage	No	2897	88.81	37
	Yes	1799	88.99	
OLB	No	6996	82.40	10
	Yes	6283	82.48	
MET	No	16542	65.66	3
	Yes	15935	65.77	
MCT	No	2808	87.66	37
	Yes	1767	88.01	

Table 13 Performance evaluation of Scenario VI (1000Tasks × 5Machines).

Algorithm	Using Trust	Makespan (sec)	Utilization (%)	Improvement (%)
Min-min	No	10394	98.01	37
	Yes	6468	97.89	
Max-min	No	14934	99.99	17
	Yes	12310	99.97	
Sufferage	No	11071	98.66	29
	Yes	7812	98.64	
OLB	No	35179	95.81	2
	Yes	34375	95.81	
MET	No	61560	79.11	1
	Yes	60860	79.11	
MCT	No	12312	98.20	28
	Yes	8812	98.28	

Table 14 Performance evaluation of Scenario VI (1000Tasks × 10Machines).

Algorithm	Using Trust	Makespan (sec)	Utilization (%)	Improvement (%)
Min-min	No	7112	95.91	42
	Yes	4111	95.55	
Max-min	No	10312	97.01	20
	Yes	8211	96.88	
Sufferage	No	7260	95.43	32
	Yes	4871	95.41	
OLB	No	25312	90.99	3
	Yes	24550	90.98	
MET	No	44179	71.48	1
	Yes	39111	71.55	
MCT	No	8542	94.88	34
	Yes	5635	94.91	

bles 10-12, Max-min algorithm is also the best algorithm in introducing good utilization; 99.64%, 96.62%, and 91.76% in Scenario IV, Scenario V, and Scenario VI respectively. MET algorithm is the worst algorithm in introducing good utilization; 65.77% in Table 12. It makes waiting time of tasks is very large and don't take ready time of tasks so the machines is almost idle.

When number of meta-tasks is 1000 as shown in Tables 13-15 (Scenario VII, Scenario VIII, and Scenario IX), Min-min algorithm is the best algorithm in reducing makespan. It recorded 49% improvement in Scenario

Table 15 Performance evaluation of Scenario VI (1000Tasks × 15Machines).

Algorithm	Using Trust	Makespan (sec)	Utilization (%)	Improvement (%)
Min-min	No	4197	90.84	49
	Yes	2100	90.66	
Max-min	No	6567	93.11	23
	Yes	5055	93.08	
Sufferage	No	4535	91.77	35
	Yes	2906	91.70	
OLB	No	16589	87.88	4
	Yes	15900	87.84	
MET	No	29780	65.91	1
	Yes	29430	65.98	
MCT	No	5156	90.02	39
	Yes	3106	90.13	

Table 16 Comparison between conventional and Proposed trust models in Min-min algorithm.

# of Tasks	Using Trust	Utilization (%)		Improvement (%)	
		Con. Model	Pro. Model	Con. Model	Pro. Model
50	No	93.17	93.19	25.28	62
	Yes	92.53	92.67		
100	No	96.15	96.15	25.32	31
	Yes	95.91	95.93		

IX. MET algorithm isn't recorded any improvement of makespan as shown in Tables 13-15. This is the only case that the proposed trust model failed in reducing makespan. MET algorithm is one type of Immediate mode heuristics, so it is static environment. It isn't suitable with the dynamic trust value changes. In the second part of the experiment as shown in Tables 13-15, Max-min algorithm is also the best algorithm in introducing good utilization; 99.97% as in Table 13 is the highest utilization value in all scenarios.

When number of meta-tasks is 50 as shown in Tables 7-9 (Scenario I, Scenario II, and Scenario III), the average improvement in all algorithms is 9%. When number of meta-tasks is 100 as shown in Tables 10-12 (Scenario IV, Scenario V, and Scenario VI), the average improvement in all algorithms is 7%. When number of meta-tasks is 1000 as shown in Tables 13-15 (Scenario VII Scenario VIII, and Scenario IX), the average improvement in all algorithms is 6 %. When number of machines is increased, the improvements of all algorithms except MET algorithm are also increased.

6.2 Comparison with Conventional Model

The complete result comparison between the conventional trust model and the proposed trust model in the three algorithms showed in Tables 16, 17, and 18.

In Table 16, the proposed trust model is better in improving the makespan than the conventional trust model in all cases. Min-min algorithm selects minimum completion time tasks with dynamic mapping of meta-tasks. So with the minimum number of tasks; 50 and 100, and the dynamic environment in changing trust values, the proposed trust model

Table 17 Comparison between conventional and Proposed trust models in Sufferage algorithm.

# of Tasks	Using Trust	Utilization (%)		Improvement (%)	
		Con. Model	Pro. Model	Con. Model	Pro. Model
50	No	94.14	94.30	32.67	56
	Yes	95.32	95.47		
100	No	97.11	97.21	33.19	30
	Yes	97.33	97.45		

Table 18 Comparison between conventional and Proposed trust models in MCT algorithm.

# of Tasks	Using Trust	Utilization (%)		Improvement (%)	
		Con. Model	Pro. Model	Con. Model	Pro. Model
50	No	93.90	93.91	34.44	45
	Yes	93.96	93.99		
100	No	96.51	96.95	34.26	30
	Yes	96.81	96.99		

can make better improvement.

In Table 17, the proposed trust model is better in improving the makespan than the conventional trust model in only one case 50 tasks. It failed to improve 100 tasks more than the conventional trust model. This result is because sufferage algorithm depends on the calculation of the minimum and second minimum completion time in every mapping of the tasks. This sufferage value takes more time beside the time takes to change the trust value. When number of tasks is increased in this algorithm, the conventional trust model may be better than the proposed trust model.

In Table 18, the proposed trust model is better in improving the makespan than the conventional trust model in only one case 50 tasks. It failed to improve 100 tasks more than the conventional trust model. MCT algorithm is one type of Immediate mode heuristics. It is static in mapping meta-tasks. The proposed trust model depends on the dynamic environment of changing trust values. So the improvement of MCT algorithm isn't always good. Except of that, when number of tasks is small as 50 tasks, the average of tasks executed in every machine will be from 6 to 10 tasks. This is small number, so the trust locating, trust computing, and trust updating will not be increased, and makespan can be reduced by the proposed trust model.

In the summary, there are six cases that compare the conventional trust model and the proposed trust model. According to the result, the proposed trust model shows better performance in four cases than the conventional trust model.

The conventional trust model and proposed trust model introduce approximating results in the utilization in the three algorithms.

7. Conclusions and Future Work

Trust and security are not the same areas in the domain of Grid. In this paper, we have proposed trust model to examine the trust in resource management systems. We have tested it in six heuristic algorithms to evaluate the performance. In the experiment, we have measured makespan and utilization. The proposed trust model can successfully reduce makespan for most algorithms, but in the utilization it isn't fully successed. The result says that our trust model provides better performance than the original trust model in most cases especially in the Min-min algorithm.

For future work, the trust model should take into consideration the failure to improve the performance. The performance evaluation of the proposed trust model with the scheduling heuristics should be improved with other large scale tasks such as millions tasks and more.

Acknowledgments

The authors are grateful to Prof. Hideo Ito for his discussions and advices.

References

[1] F. Azzedin and M. Maheswaran, "Towards trust-aware resource management in grid computing systems," First IEEE International Workshop on Security and Grid Computing, pp.452–457, 2002.
[2] F. Azzedin and M. Maheswaran, "Evolving and managing trust in grid computing systems," IEEE Canadian Conference on Electrical & Computer Engineering (CCECE '02), pp.1424–1429, 2002.
[3] F. Azzedin and M. Maheswaran, "Integrating trust into grid resource management systems," 2002 International Conference on Parallel Processing, pp.47–54, 2002.
[4] F. Azzedin and M. Maheswaran, "Trust modeling for peer-to-peer based computing systems," 12th IEEE Heterogeneous Computing Workshop (HCW 2003)(in conjunction with IPDPS 2003), 2003.
[5] F. Azzedin and M. Maheswaran, "Trust brokering system and its application to resource management in public-resource grids," 2004 International Parallel and Distributed Processing Symposium (IPDPS 2004), 2004.
[6] M. Maheswaran, S. Ali, H.J. Siegel, D.A. Hensgen, and R.F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," J. Parallel Distrib. Comput., vol.59, no.2, pp.107–131, 1999.
[7] H. Izakian, A. Abraham, and V. Snasel, "Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments," The 2009 IEEE International Workshop on HPC and Grid Applications, (IWHGA2009), 2009.
[8] B. Ma, J. Sun, and C. Yu, "Reputation-based trust model in grid security system," Journal of Communication and Computer, USA, vol.3, no.8 (Serial no.21), pp.41–46, 2006.
[9] A. Chakrabarti, Grid computing security, 1st ed., pp.33–45, Springer, 2007.
[10] A. Abdul-Rahman and S. Hailes, "Supporting trust in virtual communities," 33rd IEEE Annual Hawaii International Conference on System Sciences (HICSS-33), 2000.
[11] C. Lin, V. Varadharajan, Y. Wang, and V. Pruthi, "Enhancing grid security with trust management," IEEE International Conference on Services Computing, pp.303–310, 2004.

Sherihan Abu Elenin received the B.Sc. from the faculty of computers and information, Mansoura University, Egypt in 2002. She is currently pursuing the doctoral degree at Chiba University, Chiba, Japan. Her scientific interests include parallel processing, distributed computers, cluster systems, and grid computing. She is a student member of the IEEE.

Masato Kitakami received the B.E. degree in electrical and electronic engineering, the M.E. degree in computer science, and the Dr. Eng. degree all from the Tokyo Institute of Technology, Tokyo, Japan in 1991, 1993, and 1996, respectively. He joined Department of Electrical and Electronic Engineering, Tokyo Institute of Technology in April 1996 and moved to the Department of Information and Image Sciences, Chiba University in December 1999. From April 2001 to March 2003, he is with VLSI Design and Education Canter, the University of Tokyo. He has been with Graduate School of Advanced Integration Science, Chiba University since April 2007 and is now associate professor. Dr. Kitakami received the Young Engineer Award from the IEICE in 1999. His research interests include error control coding, dependable paralell/distributed systems, error control in data compression. He is a member of the IEEE.