Dependable Workflow Scheduling in Global Grids

Mustafizur Rahman[1], Rajiv Ranjan[2], and Rajkumar Buyya[1]

[1]Grid Computing and Distributed Systems Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{mmrahman, raj}@csse.unimelb.edu.au

[2]Service Oriented Computing Research Group
School of Computer Science and Engineering
The University of New South Wales, Australia
rajiv@unsw.edu.au

Abstract

In this paper, a reputation-based Grid workflow scheduling algorithm is proposed to counter the effect of inherent unreliability and temporal characteristics of computing resources in large scale, decentralized Grid overlays. The proposed approach builds upon structured peer-to-peer indexing and overlay networking techniques to create a scalable wide-area networking of Grid sites for supporting dependable scheduling of applications. The scheduling algorithm considers reliability of a Grid resource as a statistical property, which is globally computed in the decentralized Grid overlay based on dynamic feedbacks or reputation scores assigned by individual service consumers (Grid Resource Brokers). The proposed algorithm can dynamically adapt to changing resource conditions and offer significant performance gains as compared to traditional approaches in the event of unsuccessful job execution or resource failure. We evaluate and demonstrate the feasibility of our approach through an extensive trace driven simulation. The results show that our scheduling technique can reduce the makespan up to 50% *and successfully isolate the failure-prone resources from the system.*

1 Introduction

Grid computing enables the sharing, selection, and aggregation of distributed heterogeneous resources that are under the control of different Grid sites. Federated model for inter-connecting multiple Grid sites [1], [6] offers an opportunity for every participating site to pool its local resources as part of a single, massive scale resource sharing abstraction. In a large scale federation, both Grid users (i.e. application scientists) and resources are independent and geographically distributed over many disciplines and sites. Similarly, applications (e-Science workflows, scientific simulations) combine multiple, independent, and distributed software elements such as services, real-time data and experiments.

In federated environments such as global Grids, the availability, performance, and state of resources, applications, services, and data undergo continuous changes during the life cycle of an application. Uncertainty and unreliability are facts in federated environments, which are triggered by multiple factors, including: (i) software and hardware failures as the system and application scale that lead to severe performance degradation and critical information loss; (ii) dynamism (unexpected failure) that occurs due to temporal behaviours, which should be detected and resolved at runtime to cope with changing conditions; and (iii) lack of complete global knowledge that hampers efficient decision making as regards to composition and deployment of the application elements. The existing Grid scheduling approaches and application composition techniques [12], [17] are inadequate to handle all of these uncertainties caused by the dynamic behaviours related to resources and applications in federated environments.

The aforementioned challenges are addressed in this paper by developing a novel self-managing [2] scheduling algorithm for workflow applications that takes into account the Grid site's prior performance and behaviour for facilitating opportunistic placement of ap-

plication components. The proposed scheduling algorithm is dependable (lower probability of application failure), as it is capable of dynamically adapting to the changes in system behaviour by taking into consideration the performance metrics of Grid sites (software and hardware capability, availability, failure). The dependability of a Grid site is quantified using a decentralized reputation model, which computes local and global reputation scores for a Grid site based on the feedbacks provided by the scheduling services that have previously submitted their applications to that site. In particular, this paper **contributes** the following to the state-of-the-art in the Grid scheduling paradigm :

(i) a novel Grid scheduling algorithm that aids the Grid schedulers such as resource brokers in achieving improved performance and automation through intelligent and opportunistic placement of application elements based on dependability;

(ii) a comprehensive simulation-driven analysis of the proposed approach based on realistic and well-known application failure models to capture the transient behaviours that prevails in existing Grid-based e-Science application execution environments;

(iii) a comparative evaluation that demonstrates the self-adaptability of the proposed approach in comparison to Grid environments where: (1) resource/application behaviours do not change (i.e. no failure occurs), therefore no self-management is required and, (2) transient conditions exist but runtime systems and application elements have no capability to self-adapt.

The remainder of this paper is organized as follows. In the next section, we describe the related work that are focused on dependable application scheduling and distributed reputation models. Section 3 provides a brief discussion related to key system models including federated Grid overlay and workflow application. In Section 4, we provide the overall architecture of our dependable scheduling approach. The proposed distributed reputation management for dependable scheduling is presented in Section 5. Simulation setups, performance metrics and key findings of the experiments performed are discussed in Section 6. Finally, we conclude the paper with the direction for future work.

2 Related Work

2.1 Dependable Scheduling

A recent work by Jik-Soo et al. [12] that advocates Content Addressable Network [16], DHT based dynamic propagation and load-balancing in desktop Grids, suffer from performance uncertainty and unreliability due to lack of context awareness in scheduling. A most recent proposal or reputation-driven scheduling in context of voluntary computing environments (desktop grids) has been put forward by Jason et al. [17]. They consider a centralized system model, where a central server is assigned responsibility for maintaining reliability ratings that form the basis for assigning tasks to group of voluntary nodes. Such centralized models for scheduling and reputation management [3] present serious bottleneck as regards to scalability of the system and autonomy of Grid sites. Moreover, our approach considers scheduling of workflow applications, whereas the aforementioned approaches are targeted towards bag of tasks type of application model. Currently, Grid information services [5], on which Grid schedulers [7] depend for resource selection, do not provide information regarding how the resources have performed in the recent past (performance history) or at what level they are rated by other schedulers in the system as regards to QoS satisfaction.

2.2 Distributed Reputation Models

There has been considerable amount of research work done in Peer-to-Peer (P2P) reputation systems to evaluate the trustworthiness of participating peers. These reputation systems are targeted towards P2P file sharing networks that focus on sharing and distribution of information in Internet-based environments. The PoweTrust model proposed by Zhou et al. [20], utilizes single dimensional Overlay Hashing Functions (OHFs) for: (i) aassigning score managers for peers in the system and (ii) aggregating/computing the global reputation score. These kinds of OHFs are adequate if the search for peers/resources is based on single keyword (such as file name) or where there is single ordering in search values. However, OHFs are unable to support (or support with massive overhead) searches containing multiple keywords, range queries (such as search for a Grid site that has: Linux operating system, 100 processors, Intel architecture, and reputation ≥ 0.5). The EigentTrust model [10] suggested by Kamvar et al. also suffers from aforementioned shortcomings. To overcome these limitations, in the proposed approach a d-dimensional data distribution technique [14] is applied on the overlay of peers for managing the information related to complex searches and reputation values.

3 System Models

3.1 Grid Model

The proposed scheduling algorithm utilizes the Grid-Federation [15] model in regards to distributed resource organization and Grid networking. The Grid-Federation consists of a number of Grid sites, n, with each site contributing its local resources to the federa-

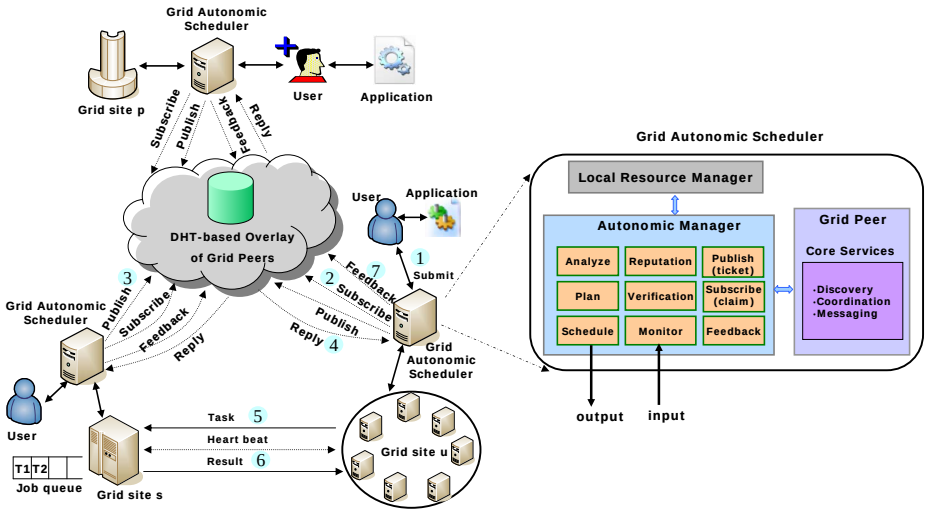


Figure 1. Reputation-based dependable scheduling architecture. Grid sites p, l, s, and u are managed by their respective Grid Autonomic Scheduler services.

tion. Every site in the federation has its own resource set descriptor that includes information about the CPU architecture, number of processors, memory size, secondary storage size, and operating system type.

The application scheduling and resource discovery in the Grid-Federation is facilitated by a specialized Grid Resource Management System (GRMS) known as Grid Autonomic Scheduler (GAS). Fig. 1 shows an example Grid-Federation resource sharing model consisting of Internet-wide distributed Grid sites. Every contributing Grid site maintains its own GAS service. A GAS service is composed of the software components: Grid Autonomic Manager (GAM), Local Resource Management System (LRMS) and Grid Peer.

The GAM component of GAS exports a Grid site to the federation and is responsible for scheduling locally submitted jobs (workflows, parallel applications) in the federation. Further, it also manages the execution of remote jobs (workflows) in conjunction with the local resource management system. The LRMS software module can be realized using systems such as SGE (Sun Grid Engine) [9].

The Grid peer implements infrastructure level core services for enabling decentralized and distributed resource discovery supporting resources status lookups and updates across a federation. It also enables decentralized inter-GAS interaction for optimizing load-balancing and distributed resource provisioing. These core services are divided into a number of sub-layers: (i) higher level services for discovery, coordination, and messaging; (ii) low level distributed indexing and data organization techniques; (iii) DHT-based self-organizing routing structure.

A Grid Peer service accepts three basic types of objects from the GAM service as regards to dependable and dynamic scheduling: (i) a *claim*, is an object sent by a GAM to the DHT overlay for locating the resources that match the user's application requirements, (ii) a *ticket*, is an update object sent by a Grid site, mentioning about the underlying resource conditions, and (iii) a *feedback*, is an object sent by a GAM regarding the reputation of a Grid site in the system upon the output arrival of a previously submitted task. In general, a Grid resource is identified by more than one attribute (such as number of processors, type of operating system, CPU speed); so a claim, ticket or feedback object is always multi-dimensional. Further, each of these objects can specify different kinds of constraints on the attribute values.

The self-organizing routing structure is largely designed over Chord [18]. Grid Peer nodes in the Chord overlay are interconnected based on a ring topology. By maintaining a small routing state of $O(log\ n)$ per node, Chord as well as other DHTs offer deterministic look ups in a completely decentralized and distributed manner. Traditionally, the basic Chord implementation is incapable of supporting complex multi-dimensional Grid resource search algorithms, as it was originally designed to support only one-dimensional search algorithms (document or name search in peer-to-peer file sharing network). In order to support complex resource discovery (processor type, OS type, CPU speed) over Chord routing structure, a multi-dimensional data distribution indexing technique [14] (a variant of MX-CIF Quad tree [19]) is implemented. In depth discussion of this aspect of the system is beyond the scope of this

paper. However, interested readers can refer to our previous work [14] for detailed information.

3.2 Application Model

In this work, we consider the Scientific workflow applications as the case study for the proposed scheduling approach. A Scientific workflow application can modeled as a Directed Acyclic Graph (DAG), where the tasks in the workflow are represented as nodes in the graph and the dependencies among the tasks are represented as the directed arcs among the nodes.

In a workflow, an entry task does not have any parent task and an exit task does not have any child task. We also assume that a child task can not be executed until all of its parent tasks are completed. At any time of scheduling, the task that has all of its parent tasks finished, is called a ready task.

Table 1. Notations: Grid, Reputation and Failure models

Symbol	Meaning
Grid	
S_i	i-th Grid site in the system
GAS_i	i-th GAS in the system
Reputation	
$succ(i,j,k)$	output of result verification function for task T_k of S_j executed by S_i.
$feed(i,j,k)^t$	feedback score of task T_k from S_j for S_i after t transactions.
NF_i	total number of negetive feedbacks given by other sites for S_i.
TF_i^t	transaction feedback value for S_i after t transactions by S_i.
$TF_{i,j}^t$	transaction feedback value from S_j for S_i after t transactions.
GR_i^t	global reputation of S_i after t transactions.
$LR_{i,j}^t$	local reputation of S_i according to S_j after t transactions.
M_{LR}	local reputation matrix.
M_{GR}	global reputation matrix.
$LR_{initial}$	initial local reputation value of each site.
$GR_{initial}$	initial global reputation value of each site.
R_{th}	reputation threshold of a site for a task to be mapped by scheduler.
$\tau_{refresh}$	time interval after which initial value is assigned to reputation score of a site.
Failure	
fp	task failing probability of a Grid site.
X_Y	failure distribution, where X% sites fail task with probability between Y and $Y + 0.1$.

3.3 Scheduling Model

This section provides a brief description of the key terminologies and the basic steps involved with the proposed scheduling approach. These steps, represented in Fig. 1 are as follows:

1. A user submits his task to the local GAS service at site S_u;

2. Following this, the GAS inserts a claim object to the DHT-based overlay to locate a *dependable* and *available* Grid site (resource) that has reasonable reputation rating (above reputation threshold) in the system;

3. The GAS, GAS_s at site S_s submits a ticket object to the overlay encapsulating the information about status (availability) of the local resource;

4. The overlay undertakes the decentralized matchmaking mechanism and discovers that the resource ticket issued by Grid site S_s matches with the resource description and reputation rating currently specified by claim object inserted by site S_u. Following that a match notification message is sent to S_u;

5. Next, GAS_u sends the task to site S_s. While the application is being processed, GAS_u periodically monitors the execution progress by sending *IsAlive* messages to S_s. IsAlive messages allow the GAS services to detect the hardware and network link failure related to the site S_s.

6. Once the execution of the task is finished, S_s returns the output to GAS_u;

7. Finally, GAS_u performs the result verification for the received output, computes the feedback score for S_s and reports to the overlay. The feedback score is aggregated to the local and global reputation scores for S_s using the proposed decentralized and distributed reputation model, described in the next section.

4 Distributed Reputation Management

In this section, the key concepts and methods related to the distributed reputation management and its application to dependable scheduling are discussed.

In a fully decentralized and distributed Grid overlay, the P2P reputation system calculates the reputation score for a Grid site S_i by considering the opinions (i.e. feedbacks) [20, 10] from all the Grid sites $\in \{S_1, S_2, \ldots, S_n\}$, who have previously interacted with S_i. After a Grid site S_j completes a transaction with another Grid site S_i, S_j provides its feedback for S_i to the overlay, which is utilized to compute the reputation of S_i. This reputation value drives the future application scheduling decision making in choosing S_i for task execution. A Grid site, which accumulates higher reputation in the system is expected to be popular in the overlay. Over the period of time, the distributed scheduling services (GASs) in the system are more likely to prefer that site in the future for placement of tasks. On the other hand, a Grid site that performs badly over a period of time would accumulate comparatively lower reputation and will eventually be shunted out of the system, i.e. would receive none or very few job submissions from the schedulers (GAS).

In the proposed approach, the overlay maintains two reputation scores for each Grid site: (i) Global Reputation (GR) and (ii) Local Reputation (LR). Here, the GAS service (on behalf of local Grid site and users) rates the Grid sites, to which it submits a task, after every successful transaction (task completion) or unsuccessful transaction (task failure) based on a feedback function, $feed(i, j, k)$. The local and global reputation scores for Grid sites are stored within the distributed overlay in the form of local and global reputation matrix. These values are recursively aggregated from the feedback scores after each transaction and utilized by the scheduling algorithm to dynamically quantify the reliability of the sites.

4.1 Feedback Generation

GAS services can use a variety of rating functions based on system consensus for computing the feedback value. Some of the example functions can include the model used by *eBay* system. The reputation scheme in eBay is simple: +1 for a good or successful transaction, −1 for a poor or failed feedback, and 0 for a neutral or don't-care feedback. In this model, the feedback score has three discrete values, which evaluate the result of a transaction. However, this model does not incorporate different types of behaviour of the participating entities (e.g. an entity is failing transactions of only a particular entity, an entity is failing transactions only at the beginning or an entity is generating successful and unsuccessful transactions alternately) into the feedback score, which is required to be considered in case of heterogeneous and dynamic resource sharing Grid environments.

In our feedback model, the GAS service at site S_j computes the feedback, $feed(i, j, k)$ for a Grid site S_i dynamically after each transaction (i.e. S_i completes execution of a task T_k submitted by S_j). First, S_j verifies the output of a task returned by S_i using the result verification function $success(i, j, k)$ that assigns a value $\in \{0,1\}$, where 0 represents an unsuccessful/failed task execution and 1 represents a successful task execution. A task execution may fail for various reasons (e.g. the resource does not have appropriate libraries installed, executables are outdated or resource has been restarted before sending all the output files). The result verification function is represented as,

$$success(i,j,k) = \begin{cases} 1 & \textit{if task execution is sucessful} \\ 0 & \textit{if task execution is failed} \end{cases} \quad (1)$$

Then S_j generates the feedback score based on the value assigned by result verification function. If the assigned value is 1, feedback score is 1; on the other hand if the assigned valued is 0 then the feedback score is calculated from an exponential distribution. The output given by the exponential function is varied over the number of failed transactions between the corresponding two Grid sites. The objective of using this exponential function is to give a Grid site greater opportunity to execute tasks at the beginning so that it is not shunted out of the system after only few failed transactions. However, if a site continues to fail more transactions, the value for exponential function approaches 0. Thus, if $F_{i,j}$ is the number of unsuccessful task executions by S_i with S_j, the feedback score for task T_k, after t transactions by S_i with S_j can be represented as,

$$feed(i,j,k)^t = \begin{cases} 1 & \textit{if success(i,j,k) = 1} \\ \alpha_f^{\frac{1}{\beta_f}^{F_{i,j}}} & \textit{if success(i,j,k) = 0} \end{cases} \quad (2)$$

where, $0 < \alpha_f \leq 0.5$ and $\beta_f \in \{1, 2, 3\}$.

If the feedback score given by a Grid site S_j is 1, we consider it as Positive Feedback (PF), whereas a Negative Feedback (NF) is attained if feedback score is less than 1.

4.2 Global Reputation Calculation

The Global Reputation (GR) of a site is a statistical reputation that is calculated by averaging all the feedbacks given by the GAS services of other Grid sites for their tasks executed at that site. Once the overlay receives a feedback, it computes the Transaction Feedback (TF) for that feedback. The value of TF depends on whether the feedback is positive or negative. If negetive feedback is received, TF is same as the feedback value. However, if feedback is positive, the value of TF is computed from an exponential distribution, where the output value is varied over the total number of negative feedbacks received by the corresponding Grid site. The purpose of using this distribution is to allow a Grid site to accrue a higher value of GR only if it executes more successful tasks than failed tasks. So, if it fails very few transactions, the output of the exponential function reaches 1 accordingly. Thus, if NF_i is the total number of negative feedbacks given by other sites for S_i, the transaction feedback value after t transactions by S_i can be calculated as,

$$TF_i^t = \begin{cases} feed(i,j,k)^t & \textit{if negative feedback} \\ \{(1-\alpha_p)+\alpha_p^{NF_i^{\frac{1}{\beta_p}}}\} \times feed(i,j,k)^t & \textit{if positive feedback} \end{cases}$$

where, $0.5 \leq \alpha_p < 1.0$ and $\beta_p \in \{4, 5, 6\}$.

The GR of a particular Grid site is calculated by taking the average of aggregated TFs form other sites. Initially GR is assigned a value $GR_{initial}$ that is greater than or equal to the reputation threshold R_{th}. After-

wards, it is dynamically changed based on the TF computed after every transaction. Thus, GR of a Grid site, S_i after total t number of transactions with other sites is represented as,

$$GR_i^t = \begin{cases} GR_{initial} & if\ t = 0 \\ \frac{GR_i^{t-1} \times t + TF_i^t}{(t+1)} & if\ t > 0 \end{cases} \quad (3)$$

The GR value of each Grid site is stored in a matrix. At any instance of time, the DHT-based distributed overlay maintains $n \times 1$ global reputation matrix M_{GR} (refer to Fig. 2(a)) for all the Grid sites $S_i \in \{1,2,\ldots,n\}$ that is updated dynamically after every transaction in the system. This M_{GR} is utilized by the distributed scheduler for mapping tasks to the Grid sites based on their reputation values.

$$M_{GR} = \begin{pmatrix} 0.85 \\ 0.90 \\ 0.70 \end{pmatrix} \begin{matrix} S_1 \\ S_2 \\ S_3 \end{matrix} \qquad M_{LR} = \begin{matrix} & S_2 & S_2 & S_2 \\ \end{matrix} \begin{pmatrix} 0.95 & 0.82 & 0.75 \\ 0.75 & 0.98 & 0.82 \\ 0.88 & 0.56 & 0.76 \end{pmatrix} \begin{matrix} S_1 \\ S_2 \\ S_3 \end{matrix}$$

(a) Global reputation matrix

(b) Local reputation matrix

Figure 2. Reputation matrix for three Grid sites (S_1, S_2, S_3).

4.3 Local Reputation Calculation

Sometime, considering only GR of a Grid site for mapping tasks, can not guarantee dependable scheduling. For example, the resource at a site S_i may fail tasks submitted by only a particular Grid site S_j. In this case, as S_j successfully executes tasks submitted by other Grid sites, its GR is high. So, the scheduler may still map the tasks submitted by S_j to S_i. Therefore, we introduce another reputation score, Local Reputation (LR) for a Grid site.

Similar to GR, LR is calculated as an average of the feedback values except it considers feedbcks from only one Grid site. TF for computing LR also follows the same function as generating TF for GR. Therefore, if $NF_{i,j}$ is the number of negative feedbacks given by S_j for S_i after t transactions with S_i, the transaction feedback value can be calculated as,

$$TF_{i,j}^t = \begin{cases} feed(i,j,k)^t & if\ negative\ feedback \\ \{(1-\alpha_p) + \alpha_p^{NF_{i,j}^{\frac{1}{\beta_p}}}\} \times feed(i,j,k)^t & if\ positive\ feedback \end{cases}$$

where, $0.5 \leq \alpha_p < 1.0$ and $\beta_p \in \{4, 5, 6\}$.

Now, the LR of a Grid site, S_i according to S_j, after t number of transactions with S_j is represented as,

$$LR_{i,j}^t = \begin{cases} LR_{initial} & if\ t = 0 \\ \frac{LR_{i,j}^{t-1} \times t + TF_{i,j}^t}{(t+1)} & if\ t > 0 \end{cases} \quad (4)$$

The LR values of each Grid site in regards to other sites are kept in a $n \times n$ local reputation matrix M_{LR} (refer to Fig. 2(b)), which is stored in the overlay and updated dynamically after every transaction between the corresponding sites. Similar to M_{GR}, M_{LR} is also utilized by the distributed scheduler for mapping tasks to the Grid sites based on their reputation values.

5 Performance Evaluation

5.1 Simulation Setup

Our simulation infrastructure is created by combining two discrete event simulators namely *GridSim* [4], and *PlanetSim* [8]. GridSim offers a concrete base framework for simulation of different kinds of heterogeneous resources, services and application types. PlanetSim is an event-based overlay network simulator that can simulate both unstructured and structured overlays.

5.1.1 Workload Configuration

In this study, we consider fork-join workflow (see Appendix) and an example of such workflow is WIEN2K [13], which is a quantum chemistry application developed at Vienna University of Technology. In this kind of workflow, forks of tasks are created and then joined, such that there can be only one entry task and one exit task. We fix the number of tasks in a workflow at 100 during the experiments but the size of each task is randomly generated from a uniform distribution between 50000 MI (Million Instructions) to 500000 MI. Further, we assume that workflows are computation intensive. Thus, the data dependency among the tasks in the workflow is negligible.

5.1.2 Network Configuration

The experiments run a Chord overlay with 32 bit configuration (number of bits utilized to generate node and key ids). The total number of GAS/broker in the system is 64. Further, network queue message processing rate is fixed at 4000 messages per second and message queue size is fixed at 10^4.

5.1.3 Resource Claim and Ticket Injection Rate

The GASs inject the ticket objects based on the exponential inter-arrival time distribution. The injection rate (i.e. resource update query rate) for the resource tickets is every 200 seconds. At the beginning of the simulation, the resource claims for the entry tasks of all the workflows in the system are injected. Subsequently, when these tasks finish, then the resource claims for the successive tasks in the workflow are posted. This process is repeated until all the tasks in the workflow are successfully completed. Spatial extent of both re-

source claims and ticket objects lie in a 4-dimensional attribute space. These attribute dimensions include the number of processors, p_i, their speed, m_i, their architecture, x_i, and operating system type, ϕ_i. The distribution for these resource dimensions is generated by utilizing the configuration of resources that are deployed in various Grids including NorduGrid, AuverGrid, Grid5000, NaregiGrid, and SHARCNET[1].

5.1.4 Reputation Configuration

The values of the parameters for configuring the reputation based scheduling in our experiment are listed in Table 2.

Table 2. Reputation parameters

parameter	value	parameter	value
α_f	0.5	$LR_{initial}$	0.8
β_f	2.0	$GR_{initial}$	0.8
α_p	0.5	R_{th}	0.8
β_p	5.0	$\tau_{refresh}$	1000 sec

5.1.5 Failure Configuration

The Weibull distribution [11] is one of the most commonly used distributions in reliability engineering and has become a standard in reliability textbook for modeling time-dependant failure data. Therefore, in this work, we use a 2-parameter weibull distribution to determine whether a task execution is failed or successful in the system. The 2-parameter weibull distribution is generally characterized by two parameters: shape parameter, β and scale parameter, η. In our experiment, the value of β is 1.2 and η is 141 sec, which is equal to the mean execution time of a task in the system.

Based on the Weibull distribution, we generate a set of resource failure distributions, X_Y by incorporating resource failure probability, fp, where X represents the percentage of resources likely to fail tasks in the system and Y represents the probability of failure. For instance, if X is 20 and Y is 0.4, then 20% of resources in the system may fail tasks with the probability (fp) between 0.4 and 0.5. The resource failure distributions, we use in the experiment are as follows:

$\boldsymbol{X}$**_0.1**: $\mathbf{0.1 \leq fp < 0.2}$; $\boldsymbol{X}$**_0.3**: $\mathbf{0.3 \leq fp < 0.4}$
$\boldsymbol{X}$**_0.5**: $\mathbf{0.5 \leq fp < 0.6}$; $\boldsymbol{X}$**_0.7**: $\mathbf{0.7 \leq fp < 0.8}$
$\boldsymbol{X}$**_0.9**: $\mathbf{0.9 \leq fp < 1.0}$

5.2 Performance Metrics

As a measurement of scheduling performance, we evaluate the following performance metrics:

Scheduling Efficiency: In order to determine the scheduling efficiency, we measure two values of the system: (i) *average makespan per workflow* and (ii) *total number of tasks failed* by all Grid sites in the system. Makespan is calculated as the response time of a whole workflow, which is equal to the difference between the submission time of the entry task in a workflow and the output arrival time of the exit task in that workflow. The measurement of makespan is taken by averaging over all the workflows in the system.

Scheduling complexity: It is expressed as the *total number of tasks scheduled* by each GAS in the system.

Pruning Efficiency: We consider pruning efficiency as the degree to which the failure-prone resources are shunted out of the system. We have measured *total number of tasks successfully executed and failed* by the resource at each Grid site in order to show the pruning efficiency.

5.3 Results and Observations

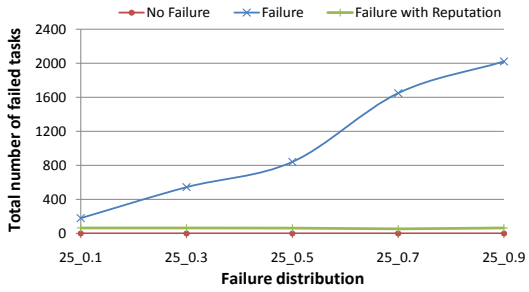
In this section, we present the experimental results obtained by simulating our reputation based dependable workflow scheduling approach and compare these with that of other approaches. The experiments are conducted with the aim at characterizing:

(i) the performance of proposed reputation based dependable scheduling approach (*Failure with Reputation*), compared to its alternatives, *No Failure* (resources do not fail any task) and *Failure without self-adaptation* (some resources fail tasks and scheduler uses a simple rescheduling technique) with respect to various performance metrics;

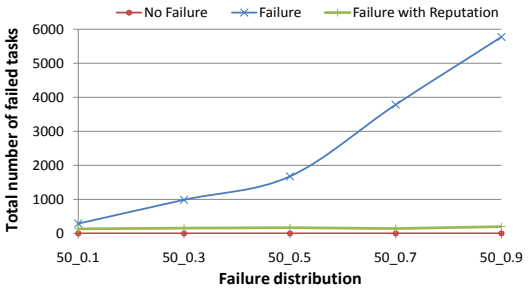
(ii) the impact of different task failure distributions on the performance of our approach and of its alternatives.

Fig. 3 presents the results of scheduling efficiency of the proposed reputation based scheduling approach against the other approaches, *Failure without self-adaptation* and *No Failure*. The total number of tasks failed by all Grid sites for each of the three approaches are depicted in Fig. 3(a) and Fig. 3(b) for different failure distributions. As we can see from Fig. 3(a) that when the failure probability of the resources is increased (for example, from **0.1** to **0.9**), the total number of failed tasks in *Failure without self-adaptation* is heavily increased accordingly. This situation is further aggravated for **50**$\boldsymbol{\_Y}$ (refer to Fig. 3(b)) since more resources are likely to fail tasks. In contrast, our approach, *Failure with Reputation* can strongly reduce the number of task failures in the system irrespective of failure distributions. This happens due to the reason that in this case, the resources with higher failure probability are not assigned any task by the schedulers as their reputation scores are decreased beyond the threshold $\boldsymbol{R_{th}}$ after few task failures. Therefore, the total number of failed tasks in *Failure with Reputation*

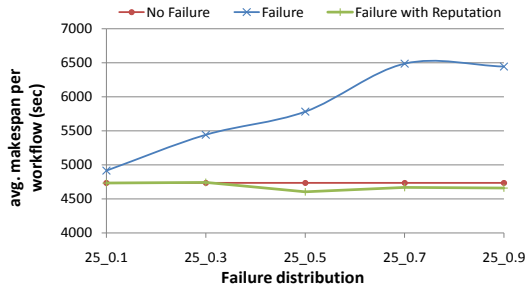
[1] http://gwa.ewi.tudelft.nl/



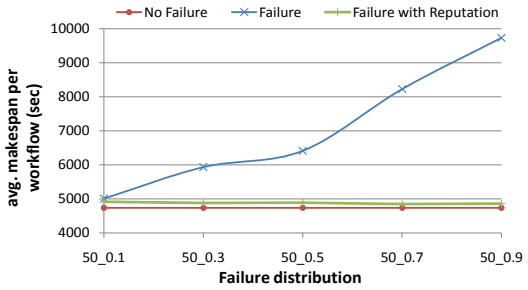
(a) Total task failures for failure distributions **25_*Y***



(b) Total task failures for failure distributions **50_*Y***



(c) Makespan for failure distributions **25_*Y***



(d) Makespan for failure distributions **50_*Y***

Figure 3. Effect of failure distribution on the makespan of workflow and the total number of task failures in the system.

is not increased with the increase in failure probability since those failure-prone resources are always shunted out of the system after few failures. For instance, total number of tasks failed by all sites in *Failure with Reputation* is upto **96.8%** and **96.5%** less than that in *Failure without self-adaptation* for **25_*Y*** and **50_*Y*** respectively.

The average makespan per workflow also shows (see Fig. 3(c) and Fig. 3(d)) similar trend (upto **28%** and **50%** makespan reduction for **25_*Y*** and **50_*Y*** respectively) as reflected in total number of task failures since if one task is failed, its child tasks can not be scheduled and eventually the completion time of the whole workflow is increased.

Fig. 4 shows the total number of tasks scheduled by GAS1 to GAS16 in the system for the failure distribution, **50_0.5**. From the figure, it is evident that in case of *Failure without self-adaptation*, each GAS needs to schedule more tasks than *No Failure* (where, GAS is not required to schedule any extra task than the size of workflow), which increases the load on the GAS accordingly. On the contrary, in case of *Failure with Reputation*, the number of tasks scheduled by each GAS in the system is almost equal to that of *No Failure* as very few tasks are failed in this approach. For example, GAS14 schedules **100** tasks in *No Failure*, **102** in *Failure with Reputation*, whereas in case of *Failure without self-adaptation*, it needs to schedule **216** tasks, which is **112%** greater than that in *Failure with Reputation* since **116** tasks, scheduled by GAS14 are failed by the Grid sites. The other GASs in the system also show the similar trend.

Fig. 5 illustrates the pruning efficiency of the proposed scheduling technique. Fig. 5(a) and Fig. 5(b) show the total number of tasks successfully executed and failed by the resources in Grid site 1 to Grid site 16 respectively for **50_0.5**. From the figures, we can realize that in *Failure without self-adaptation*, if a Grid site can execute task faster, it is assigned more tasks. Thus, the number of successful and failed tasks by that site is high if it's failure probability is low and high respectively. On the other hand, in case of *Failure with Reputation*, number of successful tasks by a Grid site is high if it is faster and does not fail any task. If it fails task, although it can execute task faster, it is not assigned any task further. Therefore, total failed tasks by that resource becomes very low. For instance, total failed tasks by resource R2 (with 0.59 failure probability and 3600 MIPS rating) is **152** in *Failure without self-adaptation*, whereas it is only **11** in *Failure with Reputation*. Fig. 5(c) shows how failure-prone resource R2 is shunted out of the system over the period of time in our proposed reputation based scheduling approach.

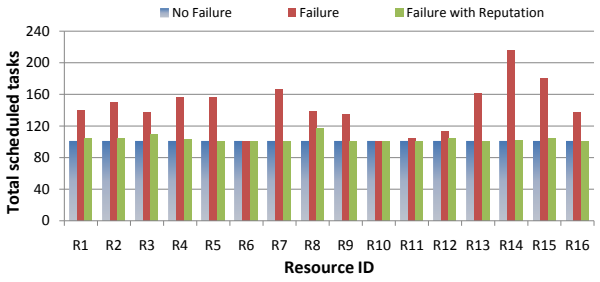
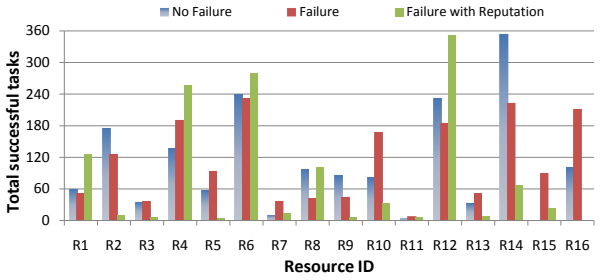
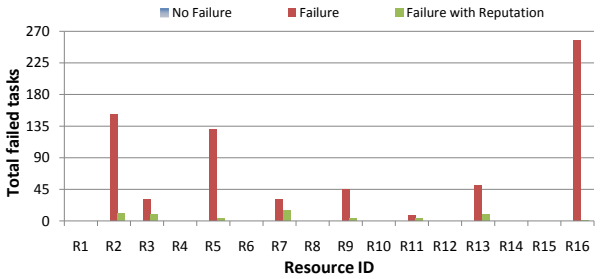


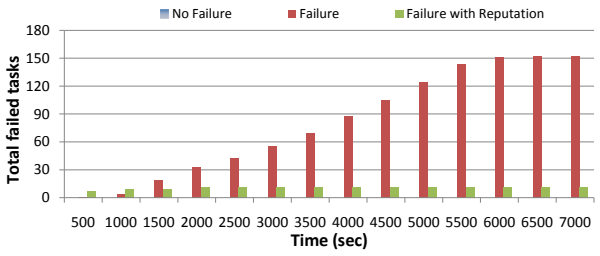
Figure 4. Total number of tasks scheduled by the GAS in the system (GAS1 - GAS16).



(a) Total number of tasks successfully executed by each resource (R1 - R16)



(b) Total number of tasks failed by each resource (R1 - R16)



(c) Total number of tasks failed by Resource **2** over time

Figure 5. Effect of considering reputation on pruning failure-prone resources.

6 Conclusion and Future Work

In this paper, we have presented a reputation based dependable scheduling technique for workflow applications in global Grids. Using simulation, we have measured the performance of the proposed scheduling technique against two cases: *Failure without self-adaptation* and *No Failure*. The results show that our scheduling technique can reduce the makespan up to **50%** and successfully isolate the failure-prone resources from the system. Thus, by applying the proposed reputation based scheduling technique, not only opportunistic placement of workflow tasks is possible but also significant performance gains are achievable (as analyzed in the previous section). Moreover, our results have practical importance since they highlight the fact that the schedulers, which do not have the ability to self-adapt in dynamic Grid conditions deliver degraded performance to application workflows.

Clearly, it is reasonable to conclude that developing self-adapting Grid scheduling and application management techniques is important to exploiting the realm of Grids. Further, adapting to dynamic resource conditions aids in coping with the unpredictability and uncertainty of Internet-scale, multi-domain global Grids. In future, we intend to evaluate and analyze the proposed approach against different workloads by varying ticket inter-arrival time and size of workflow.

7 Acknowledgements

This work is partially supported by Australian Research Council (ARC) Discovery Project grant. We gratefully thank Xiaofeng Wang for his assistance in formulating the distributed reputation model.

References

[1] *The Australian Research Collaboration Service, http://www.arcs.org.au/.*

[2] M. Agarwal, V. Bhat, H. Liu, V. Matossian, V. Putty, C. Schmidt, G. Zhang, L. Zhen, M. Parashar, B. Khargharia, and S. Hariri. AutoMate: enabling autonomic applications on the grid. In *Proceedings of Autonomic Computing Workshop*, USA, June 2003.

[3] F. Azzedin and M. Maheswaran. Integrating Trust into Grid Resource Management Systems. *Proceedings of International Conference on Parallel Processing*, 2002.

[4] R. Buyya and M. Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice*

and Experience, Volume 14, Issue 13-15, pages 1175-1220, Wiley Press, 2002.

[5] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, USA, June, 2001.

[6] Morris Riedel et al. Interoperation Scenarios of Production e-Science Infrastructures. In *OGF Workshop on eScience Highlights*, India, 2007.

[7] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor-G: A computation management agent for multi-institutional grids. In *10th IEEE International Symposium on High Performance Distributed Computing*, USA, June, 2001.

[8] P. Garca, C. Pairot, R. Mondjar, J. Pujol, H. Tejedor, and R. Rallo. Planetsim: A new overlay network simulation framework. In *Software Engineering and Middleware, SEM 2004, Linz, Austria*, pages 123–137. Lecture Notes in Computer Science (LNCS), Springer, Germany, 2005.

[9] W. Gentzsch. Sun Grid Engine: Towards Creating a Compute Power Grid. In *Proceedings of 1st IEEE International Symposium on Cluster Computing and the Grid, Brisbane, Australia*, May, 2001.

[10] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *Proceedings of 12th international conference on World Wide Web*, Hungary, 2003.

[11] D. Kececioglu. *Reliability Engineering Handbook*. Prentice Hall, Inc., New Jersey, Vol. 1, 1991.

[12] J. Kim, P. Keleher, M. Marsh, B. Bhattacharjee, and A. Sussman. Using content-addressable networks for load balancing in desktop grids. In *Proceedings of 16th international symposium on High performance distributed computing*, USA, June, 2007.

[13] D. Kvasnicka P. Blaha nad K. Schwarz, G.K.H. Madsen and J. Luitz. Wien2k - an augmented plane wave plus local orbitals program for calculating crystal properties. Technical report, Vienna University of Technology, Austria, 2001.

[14] R. Ranjan. Coordinated resource provisioning in federated grids. Technical report, The University of Melbourne, Australia, 2007.

[15] R. Ranjan, A. Harwood, and R. Buyya. A case for cooperative and incentive based coupling of distributed clusters. *Future Generation Computer Systems, Volume 24, No. 4, Pages: 280-295, Elsevier Press, Amsterdam, The Netherlands*, 2008.

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, USA, 2001.

[17] J. Sonnek, A. Chandra, and J. Weissman. Adaptive Reputation-Based Scheduling on Unreliable Distributed Infrastructures. *IEEE Transactions on Parallel and Distributed Systems, volume 18, issue 11, pages 1551-1564*, 2007.

[18] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM Conference on Applications, technologies, architectures, and protocols for computer communications*, USA, 2001.

[19] E. Tanin, A. Harwood, and H. Samet. A distributed quad-tree index for peer-to-peer settings,. In *Proceedings of the International Conference on Data Engineering*, Japan, 2005.

[20] R. Zhou and K. Hwang. PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. *IEEE Transactions on Parallel and Distributed Systems, volume 18, issue 4, pages 460-473*, 2007.