Policy-Based Support for Mobile Grid Services

Tariq Alwada'n[1], Thair khdour[2], Helge Janicke[3], Abdulsalam Alarabeyyat[2], Abdel Rahman Alkharabsheh[1]

[1]Faculty of Technology, The World Islamic Sciences and Education University
Amman, Jordan
[1]{taiq.alwadan,ar.karabsheh}@wise.edu.jo

[2]Prince Abdulah Ben Ghazi Faculty of Information Technology, Al- Balqa Applied University
Amman, Jordan
[2]{khdour,alarabeyat}@bau.edu.jo

[3]Faculty of Technology, De Montfort University
Leicester, UK
[3]{heljanic}@dmu.ac.uk

Abstract

In a multi-organization environment like the GRID, each institute might want to apply some boundaries on how its resources are being utilized by other institutes. A disagreement between the multi-Virtual Organizations (VOs) might happen in the security aspect for the policy framework. Mobile Grid Services has given the ability to move jobs, data and application software from nodes to nodes during jobs' execution in the grid environment. It has also solved some of the lack in finding suitable resources for the jobs. To facilitate the ability to support mobile resource sharing between multiple heterogeneous VOs, an authorization policy management framework is needed to support authorization for heterogeneous authorization systems. Traditional authorization policy management frameworks act well in authorization policy for a single VO where the contributing hosts grant the permission to follow a global authorization system. However most of policy management tools do not provide a clear support for sharing mobile resources between multiple heterogeneous VOs. To solve this problem, we present a dynamic and heterogeneous policy management framework that can give a clear policy definition about the ability to move jobs, data and application software from nodes to nodes during jobs' execution in the grid environment. We introduce an architecture for policy based resource management in the case of mobile sharing, and a scenario that explain the advantages of mobility mechanism and the role of policy in the grid systems. To check the performance of this architecture, a set of experiments had conducted. The results, analysis and the overheads estimation are presented in this journal.

Keywords: Grid Computing, Mobility, Policy.

1 INTRODUCTION

Due to the advances in communication technology and global system of interconnected computer networks (internet), grid computing appears as a result of a combination of multi-network computer system to develop a wide range and heterogeneous system used to solve scientific or industrial problems [1]. A grid is a system that should have the ability to organize resources[1] which are not under the subject of centralized domain, utilize protocols and interfaces and supply high quality of service [2]. Thus, the major advantage of grid computing is the capability to organize and share resources [3], [4]. As a result of such technology many challenges, such as finding suitable resources and reducing number of rejected jobs, stand in front of developing it. There are a lot of contributions to solve some of these challenges, for example: the mobility has solved some of the lack in finding the suitable resources for the jobs, but not a lot of attentions was given to the policy (aspect of security and privacy) in this solution.

Mobility is the ability to migrate or relocate jobs, data and application software among grid nodes. These migrations depend on the grid's users and the grid's nodes policies. Mobility facilitates the accomplishment of requirements for grid jobs as well as grid users. It also assists grid evolution, improves performance of operating applications by relocating data to the target host, therefore reducing the communication consumption and solving the load balancing issues. David G. Rosado et.al [5] described the mobility as "In the purview of Grid and Mobile Computing, Mobile Grid is an heir of the Grid, which addresses mobility issues, with the added elements of supporting mobile users and resources in a seamless, transparent, secure and efficient way [[6], [7], [8]]". Mobility can be divided into the following category: computer, personal and computational mobility. In personal mobility; the grid's user can do the job at sites remote from the actual physical hardware without having to move jobs around with them. They can launch a job in one site and move it to an other place in the world no matter what the machine type, such as web-based email accounts. Meanwhile the computer mobility is interested in moving actual computer hardware parts from one place to another, for example relocating PCs notebook and other PC parts. The last one, this paper is interested in, is the computational mobility. This type of mobility deals with the movement of software between nodes [9], [10]. Computational mobility may also be known as a control migration, data migration,

1. Resources refere to management and computing resources. For example: computer, software applications, etc.

link and object migration [11]. This type of migration allows the data and codes to migrate and execute on various systems across the network. Also it offers moving execution control and the ability to connect software elements at runtime while migrating from one system to another and back to the original system again.

Security is an essential element in grid computing. One of the important issue that research into grid environment tries to solve is how to keep distributed resources from unauthorized users and at the same time allowing the sharing of resources and the accountability for resource handling. Every resource applies its own security policy that may result in the refusal of requests for utilizing of its resources. Because of the fact that there are a lot of elements, like users and resources, contributing to the grid, security has become a critical aspect in checking the element trying to use a service (authentication), and in verifying whether this element is allowed or not, to use the service (authorization). Securing the grid therefore is vital to give confidence to both grid users and resource providers.

Policies are groups of regulations, standards and practices written by the administrators of resources about how their resources or jobs can be handled and used. Policies decide the way that a specific job should be accomplished, how security is applied in a domain and how an organization organizes, secures and distributes their resources. Depending on the Globus Toolkit [12], before the job submission, there should be many steps for authenticating the users who ask to use resources [13], [14]. However, after the authentication, there are no further resource access restrictions on how to use the resources.

The rest of the paper will be organized as follows. The next section describes the component of our grid architecture and describes each component in a separated section. In section three, we give an explanation of the grid portal as part of the grid architecture and its advantages. Section four presents our resource broker and its architecture including the suggested framework for the mobile grid policy services and a scenario that explains the advantage of mobility mechanism and the role of policy server in it.The following section introduces our simulation followed by validating this simulation and presents the results. In the last section we discuss future possibilities and conclude the paper.

2 Architecture Structure and Components

Grids depend on enhanced software that guarantees seamless communication between components nodes. It uses an effective mechanism which determines the suitable policy(s) that should be applied to achieve the best way to utilize resources in a way that guarantee privacy and security for both grid users and grid resources.

Figure(1) shows our proposed architecture. It applies Client/Server architecture since this architecture is the most favorable type in heterogeneous environments [15]. Client/Server network includes clients and servers who operate on the proper hardware and software for their jobs. There are two forms of client/server architecture; two and three-tier (multi-tier). Our architecture employs the last model which compromises of the client (grid portal) as the first tier, the resource broker as second tier and grid nodes as third tier. The following describes the functions for each one of them.

3 GRID PORTAL

A grid portal or grid interface is a virtual computing resource performing an interface on behalf of grid users to approach the grid. A portal has many features such as hiding the complexity of the grid from users via a simple interface which facilitates the classification of grid job necessities.

4 Resource Broker

The Resource Broker is one of the major grid elements. It performs significant functions in building a valuable grid environment by arranging user jobs onto grid resources to reach particular accomplishment targets, like cutting communication delays, raising the resource exploitation, reliability and distributing jobs across resources without depending on a particular resource. The main job for the broker is to discover and choose suitable resources for jobs by sending jobs input files to the resources, monitoring jobs and sending outputs to users.The resource broker presented in this paper is based on the mobility framework and isolates the user from the grid's middleware.

4.1 Resource Broker Architecture

The resource broker accepts job requirements from the portal and looks for appropriate resources that can fit these requirements. First it asks for all information about the available resources from the information service and the data information stored in the replica catalogue. Then it chooses the resources that can fit the job requirements and asks the grid policy agent about policies for those resources. According to that, the resource broker's architecture compromises of three components indices: information service, the replica catalogue and the grid policy agent.

4.2 Information service

Information service is a crucial element in grid computing. It is a directory service holding data about all the grid resources and the entire grid activated jobs operating on those resources. This information can be either dynamic or static information. The last one is for the hardware conditions and the operating system, while dynamic information related to the resources available time, the job presently running, type of application software, disk space and policies. In order to advertise their information the resource broker communicates to both resources and the information service to ask for this information.

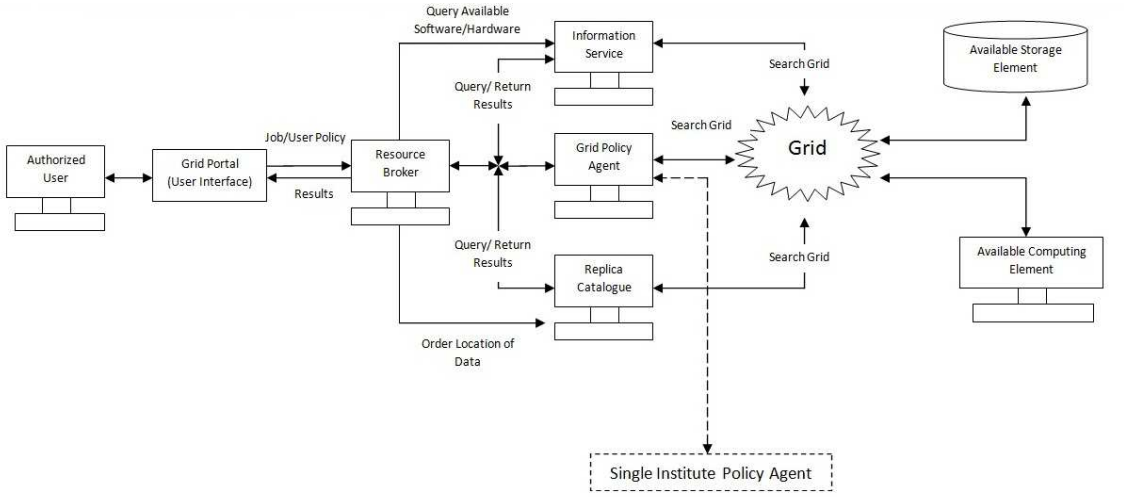

Fig. 1. Grid Architecture

4.3 Replica Catalogue

This is also an important component for the grid, because it presents information which helps in accessing the stored data in the grid. It determine the places of data in the grid, updates data resources and maps logical file names to the actual physical places on grid resources. In order to use the data on the grid the resource broker communicates with a replica catalogue to ask for information about data place and the access control needed to utlize this data.

4.4 Grid Policy Agent

The grid policy agent contains all the policies information about all resources in the grid. Each institute should have as a minimum one policy manager (agent) that has the capability to access the policy database or policy information for that institute. All policy agents (PAs) in all domains in the grid should be registered with the grid policy agent and should send their policy information (e.g. policy framework) or any changes or updated data about their policies to the grid policy agent[16]. Grid administrator can specify the policies for units participated in the grid but it does not have any policy managers (agents) that can use it directly. As an alternative, a grid policy agent operates as a proxy for the policy agents which run at each of the different institutes.

Our resource broker is differentiated from others by adding the mobility feature as a new characteristic for resource brokers. Mobility is the ability to move physical or virtual computational resources (software code, data, portable notebook PC's, running objects and mobile agents) from one site to another through a local or wide network. Mobility is a wide idea used in distributed computing. Advantages in related to services that have the capability to migrate between nodes such as increase resource utilization, enhance the organization between services and presented resources[8].

Figure(2) shows the architecture of our mobile policy agent and its components. The main job now for the institute policy agents is to merge the policies from the organization administrator and the policies from the global grid to obtain the efficient set of policies for resources belonging to that institute. The efficient set of policies are the ones applied by the policy agents attached to each resource assigned to that institute in the grid. The following describe each one of them.

4.4.1 Data/Application Software Agent

This agent is responsible for the data and application software movments. Our grid architecture lets application software and/or data to migrate from one node to another in the grid system so as to adapt the resources needed to fit the job requirements. If the resource fits the job hardware conditions and the time available, but does not have the needed application software or data needed for the job(s), the resource broker will look in the grid for the nodes that have this data/application software by checking the replica catalogue and information service and putting these nodes into a new list. Each node will be checked, one by one, by asking the mobile policy agent to decide whether or not the data/application's software policy in these nodes allows their movements or allow copying the needed software from them. The Data/Application Software Agent will check the policies for the nodes that contain the required data or application software and return the results to the resource broker. If one of the nodes does support the mobility feature for data/application software, the resource broker will copy or move (migrate) properly and send it to the resource that fits the job hardware and time requirements along with its policy. If all the nodes' policies do not support the data/application software mobility, the broker will inform the user that the grid cannot run the job.

4.4.2 Job Agent

This agent is responsible for checking the grid users' policies. Our grid architecture lets the job and its execution state to migrate from one resource to another and restart on the new one in order to fit the job conditions and requirement. If the resource that fits the job hardware conditions is occupied at the time needed, our resource broker will evacuate this resource by moving the presently operating job to other resources (if

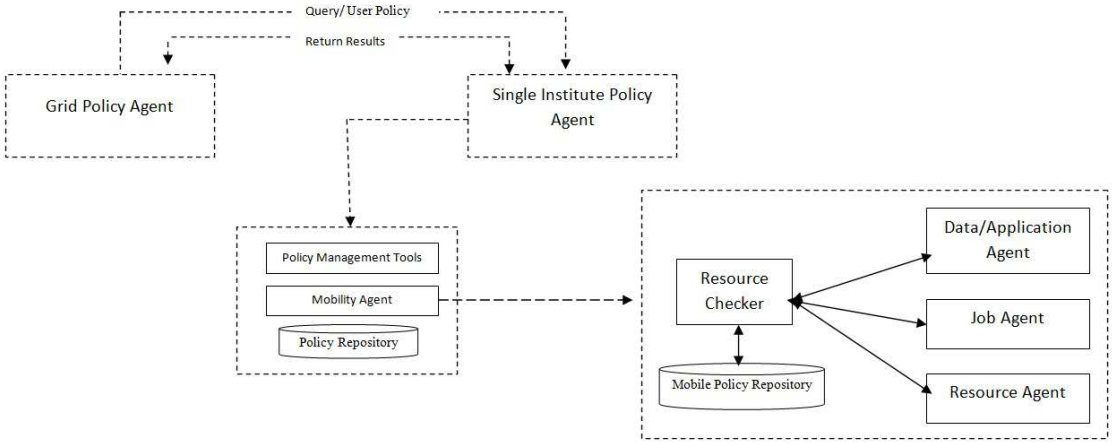

Fig. 2. Mobile agent Architecture

they exist and have the job requirements). This can be done by looking for jobs that are running on this required resource and acquire the needed information about them from the replica catalogue and the information service. If the job conditions can be fulfilled by other resources the resource broker will ask the mobile policy agent if the currently running job(s) is allowed to be migrated to another resources. The Job Agent in the mobile policy agent will check whether or not the grid user's policy allows migrating the running job to the new resource and returning the results to the resource broker. If the policy allows this kind of migration, then the resource broker will relocate these jobs to the new target resource(s) and transfer the new job to the vacated resource which can fulfil its conditions.

4.4.3 Resource Agent

This agent checks whether or not the resources' policies allow the migration for jobs, data and application software between various resources. Our grid architecture allows jobs, data and application software to move from one node to another in the grid system in order to acclimatize the resources needed to fit the job needs. If the resource that meets the job requirements is currently busy and there is a need to migirate to other resources, or there is a need for data or application software migiration, the resource broker will ask the mobile policy agent to check the policy aspect in these situations. The Resource Agent in the mobile policy agent will determine whether or not the current resource's policy allows the job migration from its node to the destination resource, or if the destination resource can accept jobs from the original resource. In both cases, it will inform the resource broker about the results. In the case of data/application software migration the resource agent in the mobile policy agent will determine if the addition or migrating of data/application software policies are allowed this type of action in the current resource and the destination resources. If they do not, the broker will inform the user that the grid cannot operate the job. If they do, the broker will apply the migration between those resources.

4.4.4 Resource Checker

As soon as the mobile policy agent makes its decisions about any possible migration(s) either for jobs, data or application's software, it stores indexes for these decision using the resource checker and stores these indexes in the policy repository prior to submitting the decision's results to the resource broker. The aim of these indexes is to track any changes or updates in the target policy(s) and inform the resource broker about them. This helps in enhancing the mobile policy agent performance and throughputs by returning to these indexes for any new requests from the resource broker instead of going for the whole checking operation again.

After the authorized grid users submit their jobs to the core of the grid system (resource broker), it asks the Grid Information Services (GIS) and Replica Catalogue about the free resources in the grid. Later, it sends this information along with the related policies (Users policies) to the Grid Policy Server which forward it to the Single Institute Policy Server to make the final Policy decisions, then it sends the results back again to Grid Policy Server. The Grid Policy Server sends the results to the resource broker to enforce the policy results in its decisions[17].

As a result the mobility has created a new environment that can solve these cases. In order to apply the mobility, the policies for the elements in Figure(3) should be checked before any migrations can take place. In our model mobile policy agent plays a significant role to achieve these requirements. The mobile policy agent checks the policies for each element in the three levels (Figure3). Each element in these levels is consider as a Policy Decision Point (PDP) where the policy decision is taken place and forward this decision to its related agent. If the policies in (level 1) have given the green light for the migration, the mobile policy agent checks the policy for the target domain (Level 2) to see if that domain is allowed to have all (or any) of the elements in (Level 1). If so, the next step should be checking the node which is going to be the new host for the elements in (Level 1). By checking the policies for both elements in (Level 1) and node's policy in (Level 3) mobile policy agent takes the decision if that node allows to have the immigrant element in (Level 1) or not.

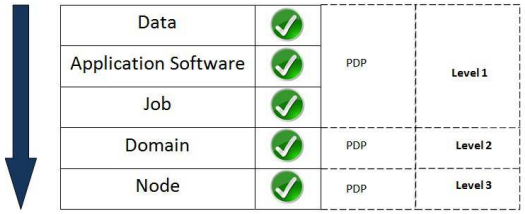

Fig. 3. Policy's Level

4.5 A Mobile Policy Agent Example

The following scenario explains the advantage of mobility mechanism and the role of policy in it within grid systems. It is divided into three sections.

- **First section: Grid Resources Specifications**
  The grid contains five nodes; each node has different conditions and specifications. These specifications are: hardware, domain, application software, data and policies. Each node is responsible for defining its own policy. Also it contains the running jobs, if presented, as shown in Tables (1), (2) and (3).
- **The second section: Jobs Requirements**
  There are five jobs which need to be executed by the grid resources. The requirements needed to accomplish the jobs include hardware, software, input, output, dmoanin and policies, as shown in Tables (4), (5) and (6). Grid users are responsible for defining their policies when submitting their jobs to the grid.
- **The third section: Fits the Jobs Requirement to Grid Resources**
  The resource broker is responsible for locating the optimal resource that can meet the job requirements and scheduling the jobs into grid resources with respect to the policies. All of these issues will be illustrated in the following. It is also shown in Figures (4) and (5).

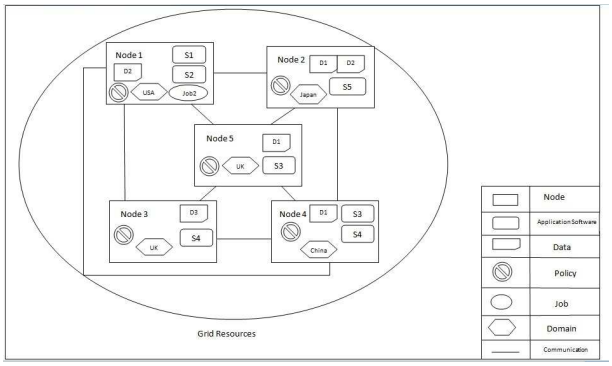

Fig. 4. Grid Resources (Infrastructure)

- **Job Migration**
  From Tables (4, 5 and 6) it can be noticed that Job1 requirements fit the Node1 specification in Tables (1, 2, 3), but Node1's policy, Table (3), is to allow only a single job to run at any time (exclusive execution), so there is a need to migrate the existing job (Job 2) on Node1 to another node that fits Job2 requirements. The resource broker looks for this substitute node and finds Node4 and Node5; but Node4 domain is in China which is against the policy of Job2 and Node1 policy. Therefore, the resource broker sends job1 to Node1 and move Job2 together with its status (memory image) to Node5 for execution.
- **Data Migration (case 1)**
  In Tables (1, 2, 3), Job3's requirements fit Node3's specifications in Tables (4, 5 and 6), but Node3 does not contain data (D2); this data is available in Node1 and Node2, Node1 policy is to allow movement of this data as well as Node3's data requirements, while Node2 is not. The resource broker will therefore send a message to Node3 telling it to take data (D2) along with its policy from Node1 and execute Job3.
- **Data Migration (case 2)**
  In Tables (1, 2, 3), Job4's requirements fit Node4's specifications in Tables (4, 5 and 6), but Node4 does not contain data (D2); which is available in Node2 and Node3(after migration). Node2 policy is not to allow movement of data to China domain, but the policy in Node3 allows this kind of movements, but the data in Node3 was moved originally from Node1 which its policy does not allow to move data to China domain. Therefore, the resource broker will send a message to User4 which says that the grid is unable to execute Job4, because the needed data is unavailable.

TABLE 2
Grid Nodes Application/Data Specifications.

Node Name	Application				Data	
			Requirement			
	File	Version	CPU Speed	Disk Space	File Name	Size
Node 1	S1	9.2	0.5	200	D2	1000
	S2	1.0	0.5	700		
Node 2	S5	2	1.0	300	D1/D2	900
Node 3	S4	1.0	1.0	500	D3	700
Node 4	S3	1.0	1.0	900	-	D1
	S4	5.0	1.0	250		
Node 5	S3	1.0	1.0	500	D1	700

TABLE 4
Node Specification Requirements.

User Name	Job Name	Node Specification			
		CPU		Memory	
		Speed	Count	RAM	S/D
U1	Job 1	1	1	1024	-
U2	Job 2	1	1	2048	-
U3	Job 3	2	2	2048	-
U4	Job 4	1	1	2048	-
U5	Job 5	2	2	2048	-

TABLE 1
Grid Nodes Hardware Specifications.

Node Name	Hardware				Domain
	CPU		Memory		
	Speed	Count	RAM	Shared or Disturbuted	
Node 1	1	1	1024	-	USA
Node 2	1	1	2048	-	Japan
Node 3	2	2	2048	D	UK
Node 4	1	1	2048	-	China
Node 5	1	1	2048	S	UK

TABLE 3
Grid Nodes Policy Specifications and Running Jobs

Node Name	Policy					Jobs Running
	Exclusive Execution	Move Data	Move Application	Move Job	Restricted (Domain/ User/ Job)	
N1	Yes	Yes	Yes	Yes	China/U4	Job2
N2	No	No	Yes	Yes	China	-
N3	Yes	Yes	No	Yes	Non	-
N4	Yes	Yes	Yes	Yes	Non	-
N5	Yes	Yes	Yes	Yes	Non	-

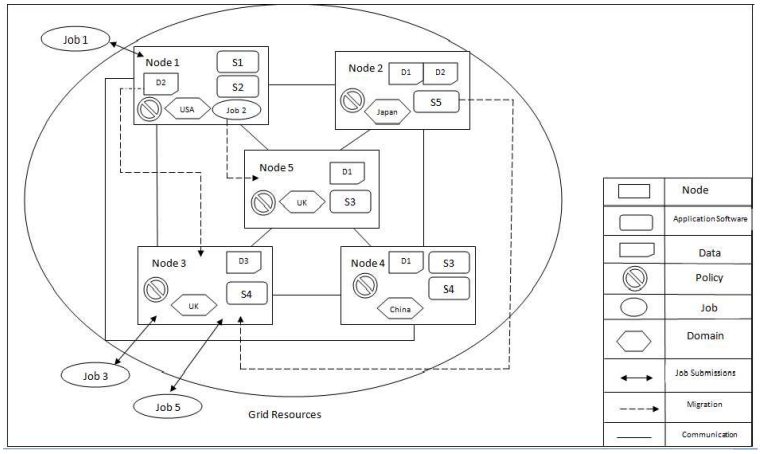

Fig. 5. Grid Resources after Mobility

TABLE 5
Application Software Requirements.

User Name	Job Name	Application Software		Data
		Name	Version	
U1	Job 1	S2	1	-
U2	Job 2	S3	1	D1
U3	Job 3	S4	1	D2
U4	Job 4	S4	5	D2
U5	Job 5	S5	1.1	D3

TABLE 6
Job Domain/Policy.

User Name	Job Name	Policy			Domain
		Exclusive Execution	Move Job	Restricted (Domain/ User)	
U1	Job 1	No	Yes	Non	China
U2	Job 2	No	Yes	China	UK
U3	Job 3	No	No	Non	USA
U4	Job 4	No	No	Non	USA
U5	Job 5	No	No	Non	UK

- **Application Software Migration**
  In the previous Tables, it can be seen that Job5's requirements fit Node3's specifications. But Node3 does not have application software (S5). Node2 does, however, and its policy is to allow this application software as well as node3's application software requirements. The resource broker will therefore send Job5 with a message to Node3 telling it to take application software (S5) from Node2 and execute Job5.

5 SIMULATION

In our simulation design we build a heterogeneous grid environment which has an unlimited number of resources in a fully connected topology. These nodes have the ability to migrate data, application software and jobs between them. The migration depends on the grid's policy, resources' policies, and grid users' policies. We have developed a Java User Interface that can simplify our work by creating a grid environment, configuring its nodes by each with its own application software, data, policies, hardware specifications and node names and finally sending jobs to the grid system.
The grid system has been simulated by using Jade simulator, which is a software framework fully implemented in Java language and allows agents to execute tasks defined according to the agent policy. When running the simulation, the main portal interface turns up. It composes all the functions needed to configure a new grid with all of its elements as shown in Figure(6).
In this interface, the grid administrator will be able to create a new grid environment by choosing grid name, configuring grid nodes and sending jobs to the grid system. The interface will then directly pass all this information to the Jade simulator to create them.

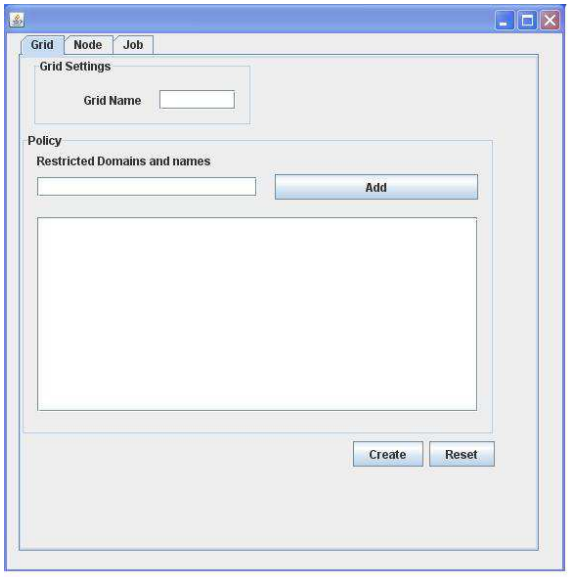

Fig. 6. Main Simulation Interface

5.1 Grid Configuration

Using the Interface in Figure(6) the grid administrator can create a new grid environment by choosing the grid name and any domains and/or users who are not allow to work under this grid. This can be done by entering the names of these domains and/or users in the Restricted Domain and names filed. This filed is going to be under the grid policy section which will be translated into XML file once the administrator clicks on the create botton, and to be sent later to Jade simulator to create the grid environment under this policy.

5.2 Node Configuration

Our interface can simulate the nodes by configuring their specifications. This is can be done by specifying their names, grid names, domain name, number of jobs that can be processed at the same time, hardware specifications, application software, data and policies. As shown in Figure(7).

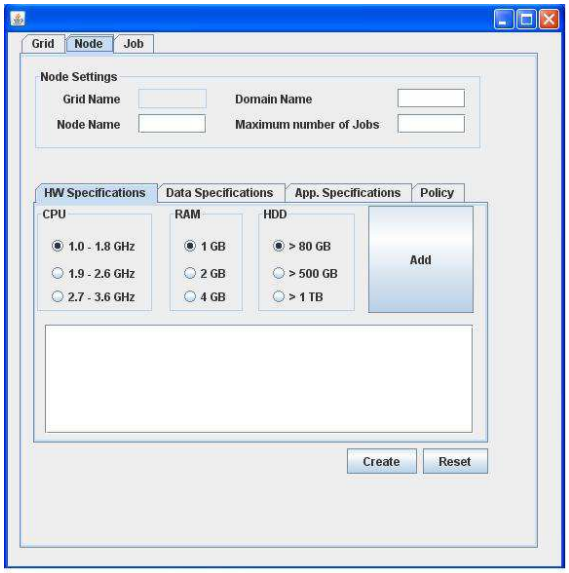

Fig. 7. Node Configuration Interface

After determining the node's name, the administrator can choose the domain name for that node. This domain name helps in sorting out the nodes into groups, which in turn will help in making the policy decision later. The administrator also has the ability to determine how many jobs each node can handle at the same time. The other fields are described as follow:

5.2.1 Node Hardware Specifications

In this step the administrator can determine the node's hardware specifications. These specifications include CPU speed, memory size and hard disk space. By choosing these specifications the system will represent the choices by an agent which can be understood by the Jade simulator and in the same time store them into the hardware section in an XML file which represents the overall node's specification. As shown in Figure(7).

5.2.2 Node Data Specifications

The interface in this part helps the administrator to configure the data in the node by determining data name and the policy for this single data. If the administrator chooses a Moving feature when creating a new data, the policy for this single data will allow this data to be moved wherever it is allowed to be moved. Otherwise it will prevent it from moving from its original node. The Copy feature has the same job but it will perform a copy action for the data instead of moving.
After finishing creating node's data, the system will represent each single data by an agent which can be understood by the Jade simulator and at the same time store the policy for this single data into the data section in the XML file which represents the overall node's specification.

5.2.3 Node Application Software Specifications

The interface in this section helps the administrator to configure the application software's in the node by determining application name and the policy for this single application. If the administrator chooses a Moving feature when creating new application software, the policy for this single application will allow this application to be moved wherever it is allowed to be moved. Otherwise it will prevent it from moving from its original node. The Copy feature has the same job but it will perform a copy action for the application instead of moving.
After creating node's application software, the system will represent each single application by an agent which can be understood by the Jade simulator and at the same time store the policy for this single application into the application software section in the XML file which represents the overall node's specification.

5.2.4 Node Policy Specifications

The interface related to the policy specifications helps the administrator to configure the policy in the node by determining any restricted domain(s) or user(s) whom they are not allowed to work under this node. Also in this section the administrator can determine whether this node is allowed to execute two jobs (or more) at the same time or not. This feature can be applied using the Exclusive choice in the policy. By choosing this option, the node is not allowed to execute more than one job at the same time.
After creating the node's policy, the system will store the policy for this node into the policy section in the XML file which represents the overall node's specification.

5.3 Job Configuration

After configuring the grid with its components by the administrator, this environment will be ready to receive jobs which have been submitted by the authorized users via the grid interface. Figure(8) shows this interface which will help the users to describe their jobs requirements in a simple way. These requirements will then be converted to a language that can be understood by the Jade simulator in the system, and at the same time it will be stored in an XML file that describes the jobs with its policies.
Our interface can simulate the jobs by configuring their

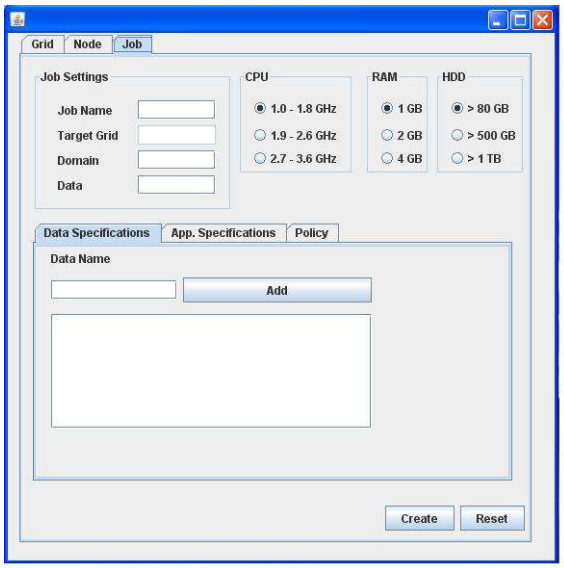

Fig. 8. Job Configuration Interface

requirements. This is can be done by specifying their names, grid names, domain name, any data attached with the job, hardware specifications, application software, data and policies. As shown in Figure(8).
After determining the job's name, the administrator can choose the domain name for that job. This domain name helps in sorting out the jobs into groups, which will help in sending the jobs to the appropriate domain later. The administrator also has the ability to attach a specific data along with the job to be processed during the execution time to fulfil the job requirements. The other fields are described as follow:

5.3.1 Job Hardware Specifications

In this step the administrator can determine the job's hardware specifications. These specifications include CPU speed, memory size and hard disk space. By choosing these specifications the Interface will represent the choosen ones by an agent which can be understood by the Jade simulator and at the same time store them into the hardware section in an XML file which represents the overall job's specification. As shown in Figure(8).

5.3.2 Job Data Specifications

By using Data interface section the administrator can configure the data needed to process the job by the grid nodes. By determining this name the system will add this data to the job requirements which will be sent later to the Jade simulator to find the suitable node(s) that owns this data. At the same time the system will store the name of this data in XML file that describes the job requirements.

5.3.3 Job Application Software Specifications

By using Application Software interface section the administrator can configure the application software(s) needed to process the job by the grid nodes. By determining this name(s) the system will add it to the job requirements which will be

sent later to the Jade simulator to find the suitable node(s) that owns this application(s). At the same time the system will store the name of this application(s) in XML file that describes the job requirements.

5.3.4 Job Policy Specifications

This interface helps the administrator to configure the job's policy before submitting it to the grid environment. The administrator or the grid users can determine any restricted domain(s) or user(s) who are not allowed to handle their jobs. Also in this section the administrator can determine whether this job is allowed to execute with other jobs at the same time or not. This feature can be applied using the Exclusive choice in the policy. By choosing this option, the job is not allowed to execute with other jobs. The Moving feature allows the job to be moved wherever it is allowed to be moved. Otherwise it will prevent it from moving from one node to another. By choosing this feature the system will store the mobility feature in the policy section in the XML file that presents the job's specifications.
After creating the job's policy, the system will store the policy for this job into the policy section in the XML file which represents the overall job's specification.

6 VALIDATION

We applied various grid environments in our simulation with numerous nodes and jobs. Each one of them with different hardware specifications, data, application software and polices. Our aim is to simulate and analyze the effect of the policy on the resource mobility (jobs, data and application software) operations in the grid environment.
Our program allows job, data and application software to migrate from one node to another in the grid environment. The aim of the simulation is to present the effect of the policies on the number of rejected jobs and number of nodes used in the grid during these migrations. To accomplish this aim, we constructed a grid environment that contains 20 nodes; each node has distinctive (or similar) hardware, application software and data specifications from others. Afterwoards we sent 30 jobs sequentially to this environment. Then we applied the job and resource mobility within the grid according to the following scenarios and configurations:

- Case 1: No mobility. In this stage we configured the policies for all of the jobs, data and application software not to be allowed to migrate within the grid along with preventing grid's nodes to accept migration resources between them. We then sent 30 jobs sequentially to the grid with distinctive (or similar) hardware, application software and data needed to accomplish these jobs. Figures (9 and 10) show the effect of polices on number of rejected jobs and number of the overall nodes used in the grid in the case of no mobility.
- Case 2: Partial Mobility (%25). In this stage we configured quarter of the policies for the jobs, nodes, data and application software to be allowed to migrate within the grid. Also we configured quarter of the grid's nodes polices to accept resource migration between them. We then sent 30 jobs sequentially to the grid with distinctive (or similar) hardware, application software and data needed to accomplish these jobs. Figures (9 and 10) show the effect of polices on number of rejected jobs and number of the overall nodes used in the grid in the case of Partial Mobility (%25).
- Case 3: Partial Mobility (%50). In this stage we configured half of the policies for the jobs, nodes, data and application software to be allowed to migrate within the grid. Also we configured half of the grid's nodes polices to accept resource migration between them. We then sent 30 jobs sequentially to the grid with distinctive (or similar) hardware, application software and data needed to accomplish these jobs. Figures (9 and 10) show the effect of polices on number of rejected jobs and number of the overall nodes used in the grid in the case of Partial Mobility (%50).

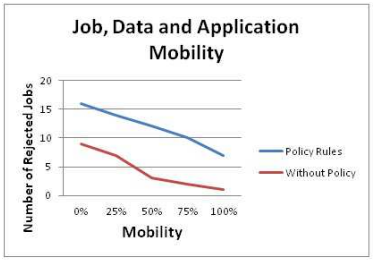

Fig. 9. Rejected Jobs with Job, Data and Application Software Mobility

- Case 4: Partial Mobility (%75). In this stage we configured (%75) of the policies for the jobs, nodes, data and application software to be allowed to migrate within the grid. Also we configured (%75) of the grid's nodes polices to accept resource migration between them. We then sent 30 jobs sequentially to the grid with distinctive (or similar) hardware, application software and data needed to accomplish these jobs. Figures (9 and 10) show the effect of polices on number of rejected jobs and number of the overall nodes used in the grid in the case of Partial Mobility (%75).
- Case 5: Full Mobility. In this stage we configured all of the policies for the jobs, nodes, data and application software to be allowed to migrate within the grid. Also we configured all of the grid's nodes polices to accept resource migration between them. We then sent 30 jobs sequentially to the grid with distinctive (or similar) hardware, application software and data needed to accomplish these jobs. Figures (9 and 10) show the effect of polices on number of rejected jobs and number of the overall nodes used in the grid in the case of full Mobility.

6.1 Rejected jobs

Once the grid is not able to accept a job due to the short of the job requirements (hardware, data and application software) amount of rejected jobs will rise. The mobility has solved this problem by migrating data or application software needed by

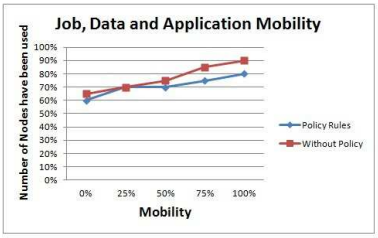

Fig. 10. The overall Used Nodes with Job,Data and Application Software Mobility

the new jobs or even evacuated the required node to fit the new jobs if necessary. Our results show the effect of the mobility on number of rejected jobs when it's applied. This means that the grid can fit a node to execute the job, consequently; the number of rejected jobs in the grid will be reduced. Nevertheless; that reducing is affected by the grid policy, node policy and job policies. It can be seen clearly that when applying these policies, the number of rejected jobs is less than the situation when the policies are not applied. In other words, the number of rejected jobs when applying the policies is less than without polices, but applying the policies over mobility gives the grid, grid's nodes and grid's user's the ability and the privacy to have control over their data, application software's and jobs.

6.2 The Overall Used Nodes

In the normal case, not all the grid's nodes own data or application software needed by all the users' jobs. In this case it can be seen clearly that some nodes are not utilized due to the short in these resources, and most of the jobs are sent to a specific nodes because of the fact that these nodes own the needed resources required by most of the jobs rather than the other nodes. In this case the gird finds itself in a situation not to accept a job, at some point, due to the short of the job requirements (hardware, data and application software) although it owns un-utilized nodes in its system. The mobility has solved this problem by migrating (or copying) data or application software needed by the new jobs, or even evacuated the required node to fit the new jobs if necessary. In this case the mobility helps in distributing user's jobs to most of the grid's nodes and that's will help in load balancing and reducing number of rejected jobs. Our results (Figure(10)) show the effect of the mobility on number of nodes used by the grid to fulfill the grid user's jobs. As a result, when applying the mobility solution, more nodes had been used than the situation without mobility.

7 CONCLUSIONS AND FUTURE WORK

We have presented in this paper a new dynamic policy management framework for mobile grid services that has the capability to deal with policies of multiple virtual organizations. Mobility assists grid evolution, improves performance of operating applications by migrating data to the execution host and therefore reduces the communication consumption and solves the load balancing problems. The other advantage of this architecture is taking the policies of the external users of the grid into account when making policy decisions. We presented our simulation for the grid environment in the case of applying grid policy, nodes' policies and users policies over mobility and finally presented the evaluation and the results of our simulation. Based on our contributions, our future work is concerned with conducting more experiments on our simulation to see the effect of the policies over mobility on the load balancing in the grid and how we can extend our simulation on the other simulators in the future.

REFERENCES

[1] I. Foster and K. Kesselman. The grid: Blueprint for a future computing infrastructure. In *Morgan Kaufmann in Computer Architecture and Design*, 1999.

[2] Alex Galis, Bernhard Plattner, Jonathan M. Smith, Spyros G. Denazis, Eckhard Moeller, Hui Guo, Cornel Klein, Joan Serrat, Jan Laarhuis, George T. Karetsos, and Chris Todd. A flexible ip active networks architecture. In *Proceedings of the Second International Working Conference on Active Networks*, pages 1–15, London, UK, 2000. Springer-Verlag.

[3] Rajkumar Buyya, David Abramson, and Jonathan Giddy. A case for economy grid architecture for service-oriented grid computing. In *Proceedings of the 15th International Parallel & Distributed Processing Symposium*, IPDPS '01, pages 83–, Washington, DC, USA, 2001. IEEE Computer Society.

[4] Zsolt N. Németh and Vaidy Sunderam. A formal framework for defining grid systems. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '02, pages 202–, Washington, DC, USA, 2002. IEEE Computer Society.

[5] D.G. Rosado, E. Fernandez-Medina, J. Lopez, and M. Piattini. Developing a secure mobile grid system through a uml extension. *Journal of Universal Computer Science*, 16(17):2333–2352, 2010.

[6] Tao Guan, Ed Zaluska, and David De Roure. A grid service infrastructure for mobile devices. In *Proceedings of the First International Conference on Semantics, Knowledge and Grid*, SKG '05, pages 42–, Washington, DC, USA, 2005. IEEE Computer Society.

[7] Hassan Jameel, Umar Kalim, Ali Sajjad, Sungyoung Lee, and Taewoong Jeon. Mobile-to-grid middleware: Bridging the gap between mobile and grid environments. In *EGC'05*, pages 932–941, 2005.

[8] J.H. Park. Usf-pas : Study on core security technologies for ubiquitous security framework. *Journal of Universal Computer Science*, 15(5):1065–1080, 2009.

[9] P. Nixon T. Walsh and S. Dobson. Review of mobility systems. In *TCD Computer Science Technical Report*, 2000.

[10] Philip W.L. Fong. Viewer's discretion: Host security in mobile code systems, 1998. School of Computing Science, Simon Fraser University.

[11] Luca Cardelli. Secure internet programming. chapter Abstractions for mobile computations, pages 51–94. Springer-Verlag, London, UK, 1999.

[12] G. Alliance. Globus toolkits.

[13] Luis Ferreira, Viktors Berstis, Jonathan Armstrong, Mike Kendzierski, Andreas Neukoetter, MasanobuTakagi, Richard Bing, Adeeb Amir, Ryo Murakawa, Olegario Hernandez, James Magowan, and Norbert Bieberstein. *Introduction to grid computing with globus*. IBM Corp., Riverton, NJ, USA, first edition, 2003.

[14] Rampure. Vishal Wu. Jin, Leangsuksun. Chokchai Box and Ong. Hong. Policy-based access control framework for grid computing. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, CCGRID '06, pages 391–394, Washington, DC, USA, 2006. IEEE Computer Society.

[15] Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design (4th Edition) (International Computer Science)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[16] Tariq Alwadan, Helge Janicke, Omer Aldabbas, and Mai Alfawair. New framework for policy support for mobile grid services. In *To appear in proceedings of the 6th International Conference on Risks and Security of Internet and Systems (CRISIS2011)*, 2011.

[17] Tariq Alwadan, Helge Janicke, Omer Aldabbas, and Hamza Aldabbas. New framework for dynamic policy management in grid environments. In *Recent Trends in Wireless and Mobile Networks, Third International Conferences, WiMo 2011 and CoNeCo 2011,*, volume 162, pages 297–304. Springer, 2011.