# Resource management system for scientific virtual laboratory applications

## Absalom E. Ezugwu*

Department of Computer Science,
Faculty of Science,
Federal University Lafia,
Lafia, Nigeria
Email: ezugwu.absalom@fulafia.edu.ng
*Corresponding author

## Seyed M. Buhari

Department of Information Technology,
Faculty of Computing and Information Technology,
King Abdulaziz University,
Jeddah, Saudi Arabia
Email: mesbukary@kau.edu.sa

## Sahalu B. Junaidu

Department of Mathematics,
Faculty of Science,
Ahmadu Bello University, Zaria,
Zaria, Nigeria
Email: sahalu@abu.edu.ng

**Abstract:** The paper presents a conceptual framework for a resource management system designed for remote virtual laboratory experimentation in both the natural and physical sciences domains. One of the key problems addressed in this paper is the use of a mathematical model to solve resource allocation or task scheduling problems in a dynamic Grid environment that consists of a heterogeneous distributed cyberinfrastructure. The main focus of this paper however includes: architectural design model for scientific virtual laboratory tasks scheduling framework, resource allocation matchmaking algorithm design, mathematical modelling of resource allocation optimisation and computational analysis of the proposed system. The research work is in line with the on-going IT infrastructure networking project at the Ahmadu Bello University, Zaria, Nigeria.

**Keywords:** grid; virtual laboratory; resource management system; scheduling; resource allocation; match making algorithm.

**Reference** to this paper should be made as follows: Ezugwu, A.E., Buhari, S.M. and Junaidu, S.B. (2015) 'Resource management system for scientific virtual laboratory applications', *Int. J. Grid and Utility Computing*, Vol. 6, No. 1, pp.8–20.

**Biographical notes:** Absalom E. Ezugwu is a Lecturer in the Department of Computer Science, Federal University Lafia, Nigeria. He is currently a PhD student at the Ahmadu Bello University, Zaria-Nigeria. He received his BSc in Mathematics with Computer Science and MSc in Computer Science degrees from Ahmadu Bello University, Zaria-Nigeria, in 2007 and 2011, respectively. His research interests include parallel and distributed computing, grid and cloud scheduling and ubiquitous computing.

Seyed M. Buhari is working with Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia. He received his BE degree in Computer Engineering from Madurai Kamaraj University, India, in 1996 and ME degree in Computer Science and Engineering from Bharathiar University, India in 1998. He has obtained his PhD in Information Technology from Multimedia University, Malaysia. His current research interests are in the areas of grid computing, IPv6 performance testing and high performance computing.

Sahalu B. Junaidu currently works as a Professor in Ahmadu Bello University, Zaria, Nigeria. Before then he worked at King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia and at Universiti Telekom Malaysia. Professor Junaidu obtained his BSc (Mathematics with

Computer Science) from Ahmadu Bello University, Zaria, Nigeria. He obtained his MSc (Formal Methods in Software Engineering) from Queen Mary & Westfield College University of London and PhD (Computer Science) from St. Andrews University, Scoltland. His research interests include high performance computing.

## 1 Introduction

One of the most demanding tasks in Grid computing is the case of resource allocation to application workloads, that is, the process of mapping jobs to distributed heterogeneous resources (Berstis, 2002; Moreno and Alonso-Conde, 2004). By resources, we mean those computing (processing nodes) and non-computing systems involved in the scheduling process. Similarly, by jobs we mean any user's experimental tasks that are meant to be executed by the available resource. Another major difficulty that is associated with resource allocation in the Grid is the lack of exact resource status information and proper documenting of log history for every scheduling event. However, researchers have proposed several efficient scheduling algorithms and computational models that are used in the Grid resources with special emphases on job scheduling. See Shojafar et al. (2010), Bu et al. (2008), Gao et al. (2005), Izakian et al. (2009) and Ramakrishan et al. (2009).

With the current advancement in the Grid computing domain, many research institutions such as universities and corporate organisations have shown keen interest in the Grid computing technology, as they exploit and integrate this platform into their computing infrastructures (Foster and Kesselman, 2005). Generally, Virtual laboratory is one application area that is built upon the Grid architecture (Afsarmanesh et al., 2000; Lawenda et al., 2004a; Lawenda et al., 2004b; Handschuh et al., 2009, p.31; Handschuh, 2012). The virtual laboratory also requires and depends upon geographically distributed heterogeneous computing and non-computing resources. Therefore, the Grid architecture is a natural fit for actualising the goals of developing an effective virtual laboratory with the key requirements for resource management and scheduling (Afsarmanesh et al., 2001; Weitzel, 2011).

There are a number of challenging issues associated with the basic operation of the virtual laboratory,[1] which are very similar to those mentioned above for the Grid computing platform. For running applications, resource management and job scheduling are the most crucial problems identified as the bottleneck to the smooth operation of the virtual laboratory system.

The research was conducted at the Department of Computer Science, Distributed Computing Laboratory Section, Ahmadu Bello University, Zaria, Nigeria. Based on the ongoing networking of IT infrastructure project which foresees the networking of both physical and natural science research laboratory facilities on the campus, the paper presents a framework model that handles the aspect of resource allocation and tasks scheduling platform for the natural science laboratory equipment available in the university.

The main focus of this work is to establish efficient resource scheduling modalities that will have significant influence on setting up multidisciplinary virtual experiment laboratory framework[2] that will allow taking control on many different devices. The rest of the paper is organised as follows. A survey of related work is discussed in Section 2. The proposed conceptual model for the resource management system is presented in Section 3. Section 4 introduces the scheduling mechanism and Section 5 covers the derivations of mathematical models required by the matchmaker algorithm to manipulate the processes involved in job-resource allocation techniques, while Section 6 presents experimental setup and numerical analysis results. Section 7 summarises the results and discusses future directions.

## 2 Related work

Inside distributed system environments, scheduling algorithms play an important role in deciding where to schedule incoming or already submitted application jobs. However, their influence is restricted to taking decisions and cannot actually apply them. In order to apply the scheduling decisions taken by a scheduler, a resource management system is required. The resource management system provides a set of services which vary depending on the system but usually involves taking jobs and physically assigning them to resources based on the logical assignment done by the scheduling algorithm. Inside a Grid, we can generally view the scheduling algorithm as the legislative entity and the resource management system as the executive part (Frincu, 2011). Likewise, several projects have been built which provided the required technology for managing batch jobs within a single distributed system or domain. These sections shortly discuss some of the two key projects that are closely related to the work presented in this paper.

### *2.1 Condor*

Condor (Thain et al., 2005) is a distributed computing resource management environment for high-throughput applications which harnesses idle time on managed resources and has capabilities for their sharing (Frincu, 2011). Condor provides numerous advanced functionalities such as job arrays and workflow support, check-pointing, job migration, rescheduling and fault recovery. It enables users to define resource requirements and rank resources and mechanism for transferring files to and from remote machines. In other words, it offers intra-domain resource management methods that allow users to harness multi-domain resources as if they all belong to one personal domain. A collector is responsible for information storage and listens for service advertisements. The resources are advertised by a resource agent, which periodically informs

the collector on the available services. A matchmaker agent is responsible for determining which resource advertised by the collector matches the desired job. It is also responsible for performing the scheduling of jobs inside a Condor. Although the system is being utilised for resources integration, it is actually intended to be used in smaller scale distributed system environments and thus should be seen more as a local job management system rather than a middleware for Grid.

### *2.2 Condor G*

Condor-G (Berman et al., 2005) is an extension of the distributed resource management software discussed in Section 2.1. Condor-G enables using Condor tools for submitting jobs online to Grid infrastructures. However, Condor-G was not designed to carter for job scheduling, but rather to execute jobs across remote distributed resource by using grid middleware. The matchmaking mechanism in Condor-G enhances it with the ability to make use of some available matchmaking algorithm to schedule jobs to Grid resource. In Condor-G matchmaking mechanism users describe their applications with the classAds language and submit them to the matchmaker. This also enables users to describe custom attributes for jobs and resources. Some of the major disadvantages (Frey et al., 2002) of Condor-G Matchmaking are lack of support for parallel jobs, lack of implicit data-aware scheduling, though users can explicitly define resources, those closer to input data are preferred. Also, in the current version, it is not possible to define a custom scheduling algorithm. Another issue associated with Condor-G Matchmaking is that it lacks integration with grid information systems. In order to use Condor-G Matchmaking, one has to develop a custom system that will provide information about resources to Condor-G Matchmaker.

In the proposed work, an attempt has been made to develop an enhanced version of job-resource matchmaking mechanism that dynamically allocates best laboratory resources to different user jobs in a distributed heterogeneous environment. Based on the empirical framework presented in this paper, it is assumed that the proposed technique performs optimally with respect to allocating parallel jobs to best matched resources. The matchmaking mechanism introduced here is very flexible as to allow for integration with the Grid information system. The backbone of the proposed resource matchmaking algorithm relies on using some mathematical concepts presented in Section 5 of this paper to perform its matching techniques.

## 3 Resource management system

Since the virtual laboratory is modelled upon the Gird architecture, there is the problem of decentralised policies in terms of distributed resource-sharing or allocation. The laboratory equipment and computational resources might be hosted across geographical locations owned by collaborating universities and research institutes. Each of these resources owners' may use different localised resource allocation strategies for their operational scenarios; therefore, the case of adopting a centralised resources allocation manager is not feasible. In line with the Grid interoperability and scalability objectives, standard mechanisms can be deployed which can be configured with appropriate localised allocation policies (Foster et al., 2001; Galstyan et al., 2005). It has also been assumed to an extent that traditional scheduling systems are often distinguished by their strategies, as embodied in algorithms and deployment parameters (Foster and Kesselman, 2003; Liu, 2007). Indications from previous work on Grid platform deployment indicates that individual system users as well as brokering intermediaries apply allocation strategies to their own jobs in addition to the traditional resource provider's, making allocation decisions for sets of jobs onto large resources (Czajkowski et al., 1998; Czajkowski et al., 2002; Venugopal, 2006).

It is therefore significant to understand what the impact of these decision will be on the performance of the entire resource utilisation for the given virtual laboratory system. The knowledge will somehow influence architectural decisions as well as scheduling strategies adopted for federated resource sharing within a scalable grid problem-solving environment.

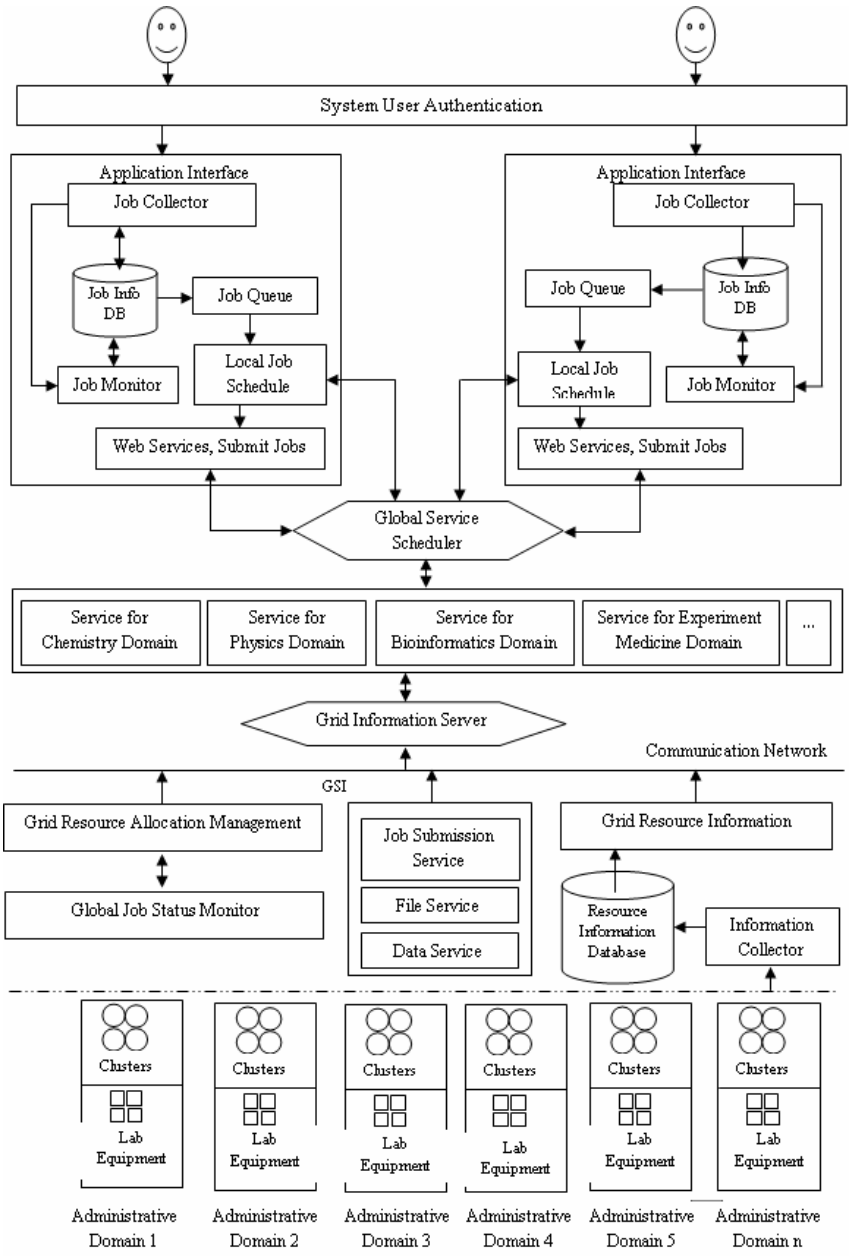
### *3.1 Model of virtual laboratory resource management system*

The Virtual Laboratory Resource Management system plays a very significant role in handling the operational activities that occurs between the different application domains and the resources located on the Grid. To construct virtual laboratories, the grid architecture is deployed. Grid computing allows for increase in the computational power due to combining of multiple resources and implementation of any application. Therefore, it is considered as a core component of the virtual Laboratory system. Figure 1 represents the structural diagram for the resource management system architecture. The system provides users with such functions as job submission, resource discovery, job management and monitoring. The scheduling model consists of job collector, scheduler, manager, information collector and database.

#### *3.1.1 Assumption about resources*

The assumption made when modelling a problem determines the variations of that problem that the model will support. We assume and incorporate two types of resources namely computational resources (such as computer clusters, servers, laboratory devices with processing ability, memory and so on) and non-computational resources (such as some laboratory apparatus lacking computing abilities, storage devices, software for experiment execution and so on). However, the distinction between computational and non-computational resources is somewhat tenuous; in some cases, non-computational resource may become computational resource. For this, it implies that the non-computational resource also possesses some computing ability or processing power similar to that of the computational resource. Some of the basic assumptions made in this paper are explained in details and presented in following section.

**Figure 1** Architecture of virtual laboratory resources management system




### *3.1.2 Assumption about tasks*

Two types of users' submitted jobs are assumed in the course of the system modelling, namely; experimental and computational tasks. For the experimental type of job, tasks are usually submitted to laboratory devices for execution, the device specifications or credentials have to be identified beforehand. While for the computational type of job, tasks are submitted to one of the application servers or computation nodes.

### *3.1.3 Job collector*

As earlier mentioned, the virtual laboratory is synonymous to a multidisciplinary problem solving environment. The nature of tasks being handled on this platform is heterogeneous and distributed in nature and encompasses different research fields and application areas. The function of the job collector is to gather the basic information about submitted jobs that are necessary for further processing and as required by the entire global scheduling process and store them in the job information log database. Some of the key job credentials required for execution include job title, location, necessary parameters for execution and the destination directory path name for the generated output file and so on. The job collector is normally a user-oriented interface linking the job monitoring module to the web service which serves as a message carrier for the global scheduler.

### *3.1.4 Grid information service*

The Grid information service integrates every other component of information services, whether static or dynamic, in the Grid system to provide a unified information access interface for

application users. Information service mainly includes two parts that comprise of resource acquisition and evaluation. Information services within the context of Grid environment provide a number of dynamic updates information such as the load, performance and so on. Some of the static information services provided by the Grid system with respect to resources deployable in virtual laboratories are: IP addresses for laboratory devices, operating system and storage capacity and other relevant information.

### *3.1.5 Administrative domain*

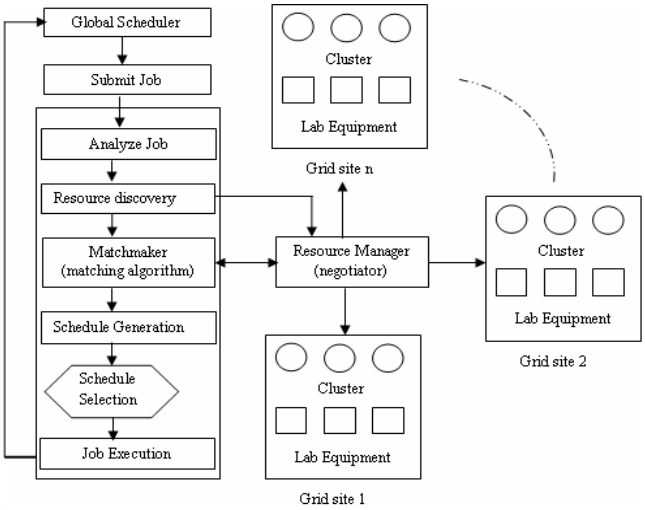
Each administrative domain is comparable with remote virtual site location capable of providing resources on demand and rendering services akin to most problem-solving environment upon the computational Grid. A virtual site may consist of multiple physical sites if they are interconnected by a high bandwidth network. Each site consists of computational servers, laboratory devices, storage systems, visualisation servers and software for experiment execution. A virtual site can be thought of as a 'regional resource centre', a composite object containing a number of data servers, processing nodes and software for laboratory experiment execution where all are connected to a local area network.

## 4 Scheduling model

The scheduler is responsible for resource discovery, resource selection and job assignment. The resource discovery algorithm interacts with an information service (the Grid Information Service), identifies the list of authorised and available machines and equipment that can execute the submitted tasks and keeps track of resource status information. The resource selection algorithm is responsible for selecting those resources that meet the required performance criteria (CPU – hours, storage capacity, processing speed, network bandwidth and so on...) along with optimisation requirements (Han et al., 2008). The scheduling model shown in Figure 2 consists of scheduler, job analyser, resource discovery, resource selection, schedule generation, schedule selection and information collector. The resource selection, schedule generation and schedule selection involves detailed information about performance capabilities and most importantly they revolve around a more informed decision-making process that provides some high level of intelligence to the operational activity of the resource management system.

It would be significant to understand those integral components that make up the building blocks of the scientific virtual laboratory; these include the system application user, scheduler, resource management centre, resources and laboratory equipment. A user first logs on to make use of the preferred domain application resident in the virtual lab, experiment is performed by the user and job is submitted to scheduler. The scheduler first analyses the job, splits it into various tasks and distributes the tasks to several resources based on the resource information collected from the resource manager. Allocation of resource is only made when user's requirement matches with the available resource profile.

**Figure 2** A structure of scheduling model




### *4.1 Job submitting*

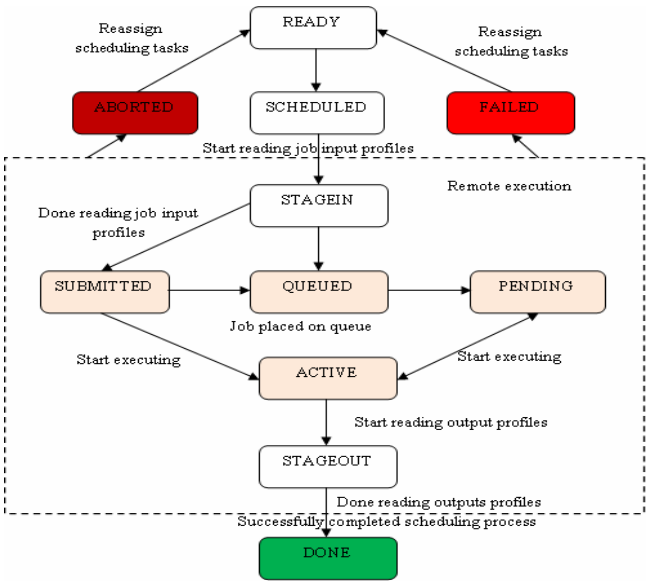
The work of the job submitting module is to schedule job to resource node in accordance with the scheduler results. Uniformity in job distribution is achieved by the scheduler via generating log files for job and transferring it to a specific resource node. Unfortunately, the execution of users' submitted jobs greatly depends on the steadiness of the local grid resource, as there are times when a job might fail to execute. The job monitoring module monitors the updating job profile information available in the job status log table; this is very important and mandatory so as to enable the system deal with any job failure execution.

In the course of job scheduling, a job passes through several states (Venugopal, 2006) as is outlined in Figure 3. A job is an input to the scheduler which allocates it to a set of resources based on its requirements profiles. The first status of any user job is the 'ready' state, after which the status is then changed to 'scheduled'. During the 'stage in' state, which is the point when the job is to be admitted into the remote scheduling process, input profiles and executables required for the job are staged to the remote resource. When this process is completed successfully and a handle is obtained, then a job is considered to be finally 'submitted'. The job may be queued while waiting for an available processor and its state changes to 'pending'. When the job starts its execution, it is considered 'active'. After the job has finished executing, it enters the 'stage out' stage where its output profiles are transferred back to the resource provider. If all its output profiles are received and are as expected by the task requirements, then the job is considered as 'done'. If any one of the state transitions fails on the remote side or the job has completed on the remote side but has not produced the expected result profiles, then it is considered 'failed' and is reassigned for rescheduling. Also if the job was interrupted and could not complete the execution process or it is terminated prematurely, then it is considered 'aborted' and is re-set for scheduling.

### *4.2 Resource manager*

The Resource Manager consists of the following components: Grid monitor, Hidden Markov Model (HMM) and resource analyser. A similar model has been introduced by Jacob et al. (2011). It often provides mediatory services among the following group of activities: requester, matchmaker and the Grid service providers. The resource manager receives request from the requester and receives the results from the matchmaker engine and gives back to the scheduler. It manages all resources in various Grid sites. The resource manager accepts the resource into the grid site only if the matchmaker ontology matches with the resource ontology (ontology is a formal explicit specification of a shared conceptualisation[3] (Gruber, 1993)). The Resource Manager acts like a registry in which the resources ontologies and locations are registered. The Resource Manager checks the request and resource ontology and directs the request to the appropriate resource providers and sends the result back to the requester.

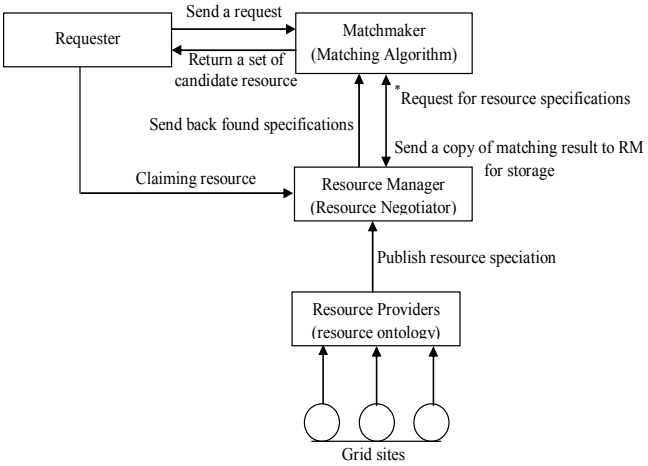
**Figure 3** Grid platform job execution phases (see online version for colours)




### *4.3 Matchmaker*

The matchmaking framework depicted in Figure 4 includes a resource specification component and matchmaking algorithms. Upon receiving a request (a request is composed using vocabulary in the request ontology) the matchmaker activates the matching algorithm to find a list of potential matches sorted according to the requester's preference criteria. The matchmaker simply returns the matched list (or *NoMatchFound*) to the requester (i.e. feedback). The matchmaker gets all resource's ontology from the Resource Manager. It matches the resource with the request by considering the static, dynamic and behavioural parameters in ontology and uses the mathematical models (presented in Section 4.) to calculate the matching degree. After calculating the matching degree, it calculates the rank for the resource by considering weight value for each parameter and forwards the result to the Resource Manager. The requester can then claim the resource by contacting the resource providers for their services via the resource manager using the claiming rule (which comprises of validly matched results ontology).

**Figure 4** Matchmaker activity framework




*A request specification includes a matchmaking function and possibly two additional constraints, a *cardinality threshold* and a *matching degree threshold*. The cardinality threshold specifies how many resources are expected to be returned by the matchmaking service. The matching degree threshold specifies the least matching degree of the resources returned by the service.

The four major steps involved in the matchmaking process are described as follows: (a) Resource providers advertise by sending their resource descriptions to the resource manager; (b) A requester sends request comprising of resource specifications to the matchmaker; (c) The matchmaker executes a matchmaking algorithm based on the requester's order by first sending request to RM to confirm the availability of the requested resource specifications, if resource available, RM sends back all the results to the matchmaker to implement matching process and returns a set of ranked resources to the requester and also stores a similar copy in RM; (d) The requester chooses the highest ranked resource from the set and then contacts the corresponding resource provider directly to claim the resource.

### *4.4 Schedule generation*

Schedule generation module performs the tasks of generating schedule marks ID containing log reference ID; indexes of selected distributed resource; IP address of selected resource; job description; job states and other related information that are required to effect schedule decision that will lead to the desired job execution. On the other hand, when a fault is detected in any of the resource nodes by the resource monitor, the process of schedule generation is re-scheduled again so as to achieve effective scheduling result. The process is repeated for $n$ number of times in as much as there is $n$ number of jobs in the job queue awaiting execution.

### 4.5 Matchmaking resource and request specifications

The set of rule for constructing the attribute names for an attribute as stated by Bai et al. (2004) are:

1 If the attribute refers to a slot of the resource class, the attribute name is the slot name.

2 If a slot of the resource class refers to an instance of other class and the attribute refers to a slot of this instance, the attribute name is the combination of the two slot names connected by "."

Following the same conceptual view of the above rule, we can construct an attribute name for a set of experimental devices/apparatus say with resource class 'microscope' in the following format:

Both requests and resources are expected to share similarity in their specification and representation syntax, either in the form of attribute-value pairs or any other acceptable format that is known to both the application users and resource providers. It is equally important to note that for the matching process to work properly, it is suggested that both the requester and resource providers use the same attribute naming convention and agree upon attribute values beforehand. Figures 5 and 6 are two examples of resource and request specifications represented in the form of attribute-value pairs. These illustrations are nothing more than abstract denotation of previous explanation of resource matching and allocation by the matchmaking framework.

**Figure 5** Example of resource instance: seven set of electron microscope

*Microscope.Name = "Electron.bio.edu"*
*Microscope.IPAddress = "microscope.electron.bio.abu.edu"*
*Microscope.resolutionSE = 3.0nm at 30kv (high vacuum mode)*
*Microscope.resolutionBSE = 4.0nm at 30kv (variable pressure mode)*
*Microscope.Magnification = "x5 ~ x300.000"*
*Microscope.AcceleratingVoltage = 0.3 – 30kv*
*Microscope.LowVacuumRange = 6 ~ 270 pa through graphic menu*
*Microscope.OperatingSystem = "windowsXP"*
*Microscope.MaximumSpecimen.Size =153mm in diameter*
*Microscope.NumberAvailable = 7*
. . .

**Figure 6** Example of request specifications

*Request.Name="user1"*
*Request.ResourceType = "Microscope"*
*Request.Resource.Name = "Electron.bio.edu"*
*Request.Resource.IPAddress = "microscope.electron.bio.abu.edu"*
*Request.Resource.resolutionSE = 3.0nm at 30kv (high vacuum mode)*
*Request.Resource.resolutionBSE = 4.0nm at 30kv (variable pressure mode)*
*Request.Resource.Magnification = "x5 ~ x300.000"*
*Request.Resource.AcceleratingVoltage = 0.3 – 30kv*
*Request.Resource.LowVacuumRange = 6 ~ 270 pa through graphic menu*
*Request.Resource.OperatingSystem = "windowsXP"*
*Request.Resource.NumberAvailable = 3*
. . .

### 4.6 Scheduling algorithms

Several implementations of the matchmaking frameworks and algorithms exist in literature; see (Kuokka and Harada, 1995; Bai et al., 2004; Liu and He, 2007; Shu et al., 2007; Han et al., 2008). Part of the services rendered by the matchmaker is to execute a matching algorithm for each request sent by the requester. The inputs to the algorithm are the request and the grid resource instances stored in the knowledge base of the matchmaking service. The matchmaking algorithm evaluates the request function in the context of each resource instance in the knowledge base. The output of the algorithm is a number of grid resources ranked according to their matching degrees. Let *n* denote the cardinality threshold specified by the request. The matchmaking algorithm returns the grid resources that have the *n* largest matching degrees to the requester. The matching degrees are computed using the mathematical models discussed in section 5. In a nutshell, these models distinctively distinguish our algorithmic framework from those found in literatures. The pseudo code of the matchmaking algorithm is shown in algorithm listing 1.

**Algorithm 1:** Pseudo-code for the match maker agent

1: **Input**: req //request specification for resource $R_i$
2: **Output**: matched list of set of candidate resource instances cr
3: **BEGIN**
4: cr = ∅
5: n = req.CardinalityThreshold;
6: mdt = req.MatchedDegreeThreshold;
7: /* Matchmaker gets corresponding resource specification
8: (ontology) for Ri from RM */
9: ***for each*** resource r in $R_i$ ***do***
10: **for** i = 1**to** k ***do*** // where k is the $k^{th}$ resource instance
11: // compute the matching degree md for resource r
12: md = compute req.RequestFunction(r[i]);
13: ***if*** (md>=mdt)***then***
14: cr[i]← r[i]; //insert r into cr
15: **update.**cr_size;
16: // cr_size is the size of the cardinality set cr
17: **endif**
18: ***if*** (cr_size > k*n)***then***
19: **break;**
20: **endif**
21: //compute the ranking expression $p_{ik}$ for the set of
22: //candidate resource instances *cr*
23: cr($r_i$ )← {$p_{i1}$, $p_{i2}$,…,$p_{ik}$ }; // store result in cr and RM
24: //we use $p_{ik}$ to denote the *k*-th candidate for the *i*-th resource in cr
25: **end for**
26: **end for**
27: **END**

The ranking process adopted for the framework considers an instance where resource that compare equally receive the same ranking number (an instance of multiple resource request by user) and the next resource(s) receive the immediately following ranking number. Equivalently, each resource's ranking number is one plus the numbers of resources ranked above it that are distinct with respect to the ranking order. Thus, if $r_1$ ranks ahead of $r_2$ and $r_3$ (which compares equally) which are both ranked ahead of $r_4$, then $r_1$ gets ranking number 1 ('first'), $r_2$ gets ranking number 2 ('joint second'), $r_3$ also gets ranking number 2 ('joint second') and $r_4$ gets ranking number 3 ('third').

## 5 Mathematical modelling

The virtual laboratory system has its own peculiar challenges when it comes to resource sharing and/or scheduling. Tasks are interactive and are scheduled according to dynamic measurement scenario. Also time slot reservation is required for experiments execution, especially for those experiments that need supervision or observation by multiple individuals online. Therefore the authors have assumed that the best approach to solve the resource management problem for virtual laboratory system is to tackle the problem by adopting several computational models. The hyper-heuristic method and hybrid algorithms are examples of where multiple techniques were merged and used to achieve quality results (Braun et al., 2003; Ritchie and Levine, 2004; Bhanu and Gopalan, 2008). In this research, we follow a similar trend by considering different mathematical models that can assist us solve the problem of multiple resource selection in the Grid system with regard to virtual laboratory application. However, the objective of introducing this concept is to devise suitable ways of achieving optimal resource selection by the scheduling system.

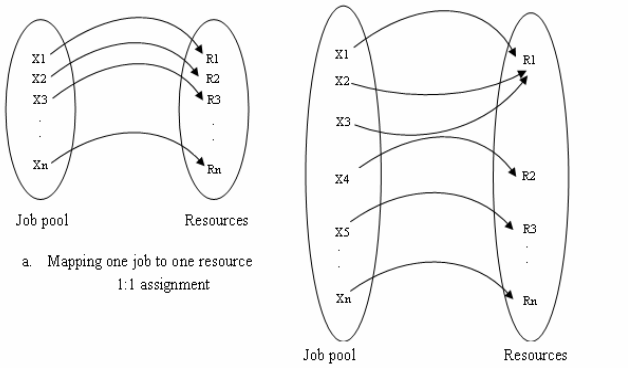
### *5.1 Mapping model*

In the system under consideration, we assume that there are $n$ distributed and shared resources $R_1$, $R_2$,…,$R_n$, comprising of compute and non-compute resources as specified by the resource requirements and $m$ independent jobs $X_1$, $X_2$,…,$X_m$ competing for the resources. We also assume that $X_i$ and $X_j$ can compete for one resource, for $i \neq j$. If we let $R = \{R_1, R_2, R_n\}$ and $X = \{X_1, X_2, X_m\}$, then we can define a non-invertible function $f$ (a function which does not have an inverse except for the case where $f = R$) from $X$ to $R$ as follows: $F{:}X \rightarrow R$. We say that two sets $X$ and $Y$ of jobs are *compatible* if $f(X) \cap f(Y) = \varphi$, otherwise, $X$ and $Y$ are incompatible. That is, resource allocation can only take place from one resource $R$ to more than one job sets if the job sets all have the same attributes. However, allocation between the resource $R$ and a job $X$ occurs if the function $F$: $X \rightarrow R$ exists. In this case, we say that $X$ is compatible with $R$. Figure 7 is a diagrammatical representation of the compatibility relation between job-resource assignments as explained earlier on.

### *5.2 Statistical modelling*

The suboptimal resource is selected by considering some basic statistical techniques such as assignment technique, correlation technique, sample testing technique and measures of dispersion. The remaining part of this section is devoted to developing statistical model to ascertain the degree of correctness of the proposed virtual laboratory resource management systems.

**Figure 7** Compatibility job to resource assignment




#### *5.2.1 Parameter (assignment technique)*

Resources usually have sets of attributes or credentials that are used to identify them on the grid. Some of these attributes are resource name, resource type (this is further categorised into compute and non-compute resource types), memory sizes, resource capacity, resource IP addresses and so on, depending on the resource providers' requirements' definition policies. If an application user provides this set of resource information alongside its submitted jobs, then the resource will be chosen among the various resources that closely match the request using assignment method. Table 1 shows job-resource assignment, while Figure 8 illustrates the job-resource assignment strategy.

**Figure 8** Job assignment strategies

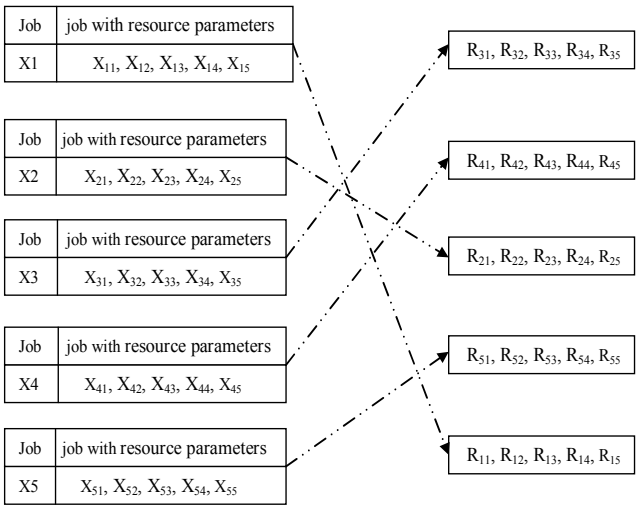


**Table 1** Job-resource assignment

| *Resource* | *Job ID* | *Resource ID* | | *Job-resource assignment* | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | *Y1* | *Y2* | *Y3* | *Y4* | *Y5* | *... Yn* |
| Name | X1 | Y1 | XI | $R_{11}$ | $R_{12}$ | $R_{13}$ | $R_{14}$ | $R_{15}$ | |
| IP address | X2 | Y2 | X2 | $R_{21}$ | $R_{22}$ | $R_{23}$ | $R_{24}$ | $R_{25}$ | |
| Memory | X3 | Y3 | X3 | $R_{31}$ | $R_{32}$ | $R_{33}$ | $R_{34}$ | $R_{35}$ | |
| Type | X4 | Y4 | X4 | $R_{41}$ | $R_{42}$ | $R_{43}$ | $R_{44}$ | $R_{45}$ | |
| Capacity | X5 | Y5 | X5 | $R_{55}$ | $R_{52}$ | $R_{53}$ | $R_{54}$ | $R_{55}$ | |
| … | … | … | … | | | … | | | |
| … | … | … | … | | | … | | | |
| … | Xm | Yn | Xm | | | | | | … $R_{mn}$ |

### *5.2.2 Parameter (correlation technique)*

Information on the relationship between two sets of variables can be obtained by the use of correlation coefficients. Correlation coefficient shows the following:

1 Whether the relationship is positive or negative.

2 The strength of the relationship.

The correlation between the request and the resource is calculated using the following:

Let $X = \{X_1, X_2, X_3, ..., X_m\}$ represent a set of jobs, where $X_m$ is the $m$-th job.

Let $R = \{R_1, R_2, R_3, ..., R_n\}$ be the set of resources, where $R_n$ is the $n$-th resource available in the grid.

Let

$$R_1 = \{R_{11}, R_{12}, R_{13}, ..., R_{1m}\}$$
$$R_2 = \{R_{21}, R_{22}, R_{23}, ..., R_{2m}\}$$
$$R_3 = \{R_{31}, R_{32}, R_{33}, ..., R_{3m}\}$$
$$\dots$$
$$R_n = \{R_{n1}, R_{n2}, R_{n3}, ..., R_{nm}\}$$

Then

$$r_i = \frac{m\sum_{j=1}^{m} X_j R_{ij} - \left(\sum_{j=1}^{m} X_j\right)\left(\sum_{j=1}^{m} R_{ij}\right)}{\sqrt{\left[m\sum_{j=1}^{m} X_j^2 - \left(\sum_{j=1}^{m} X_j\right)^2\right]\left[m\sum_{j=1}^{m} R_{ij}^2 - \left(\sum_{j=1}^{m} R_{ij}\right)^2\right]}}$$

where $i = 1,2,\dots,n$

$m$ = total available number of resources

$\sum X_j R_{ij}$ = sum of the product of paired $j$-th job to $j$-th resource

$\sum X_j$ = sum of $i$-th jobs in the job pool

$\sum R_{ij}$ = sum of $i$-th participating resources

We use the symbol $r_i$ to stand for the correlation. Through the magic of mathematics it turns out that $R$ will always lie between –1.0 and +1.0. If the correlation is negative, we have a negative relationship; if it's positive, the relationship is positive. A negative correlation indicates poor relationship between placed request and the probability of finding suitable resources (which invariably produces a poor response time). However, a positive correlation indicates good relationship between resource request and finding suitable resource.

### *5.2.3 Parameter (sample testing technique)*

The behaviour of service is calculated by testing of samples for each parameter based on the following single regression model:

$$Y_i = \beta_1 + \beta_2 X_i + \mu_i \quad \begin{matrix}\text{Observational value of} \\ \text{the number of resources used}\end{matrix} \tag{1}$$

$$Y_i = \hat{\beta}_1 + \hat{\beta}_2 X_i + e_i \quad \begin{matrix}\text{Estimated value of the} \\ \text{number of resources used}\end{matrix} \tag{2}$$

$$\hat{Y}_i = \hat{\beta}_1 + \hat{\beta}_2 X_i \quad \begin{matrix}\text{The estimated average value} \\ \text{of the resources used}\end{matrix} \tag{3}$$

where

$X_i$ = the number of response time per second

$\beta_1$ = the actual value of the intercept term

$\beta_2$ = the actual value of the slope

$\hat{\beta}_1$ = the estimator of $\beta_1$

$\hat{\beta}_2$ = the estimator of $\beta_2$

$\mu_i$ = the stochastic term (unobservable)

$e_i$ = the error term (observable)

## 6 Experimental setup

The experiments conducted are aimed at showing the efficiency of the proposed algorithm compared to the existing one in Condor-G. The experiment was conducted with 1600 resource providers with varying number of resources. The request sent per time ranges from 1 to 100. It

was assumed that all service providers are available at all time, provided they are free with their resources. The number of requests served per second increases as the system receives more jobs. The optimal resource is selected by considering mapping model, assignment model, correlation model, testing of samples and measures of dispersion. The experimental findings and results are presented in Section 6.2.

### *6.1 Simulation tools*

The implementation of the experiments was performed on GridSim Version 4.0 simulator. GridSim is a java-based discrete event Grid simulation toolkit. This simulation software enables us to directly evaluate the practicability and test the performance of the proposed scheduling strategy presented in this paper. It also provides a comprehensive facility for simulation of different classes of heterogeneous distributed resources, users, applications, resource brokers and schedulers (Buyya and Murshed, 2002). An econometric model was likewise used to analyse the result of the mathematical models. Econometric model is a statistical model used in econometrics; the model specifies the statistical relationship that is believed to hold between the various quantities under study. To compare the relative performance of the existing Condor G matchmaker algorithm, we simulated a serial job allocation mechanism and then compared the simulation results with that of the proposed algorithm. It has been mentioned that the existing technology matches jobs to resources sequentially.

### *6.2 Experimental results and analysis of the mathematical models*

This section is a discussion on the implementation and analysis of the model presented in Section 4. Descriptive statistics and regression analysis were used to perform data analysis for entries in Table 2. Econometric model was used to analyse the effect of the average response time (sec.) on the number of resources.

**Table 2** Time taken to find the best matched resources for a job by statistical method

| *No. of Resources* | *No. of Jobs* | *Time taken in second* | | | | |
|---|---|---|---|---|---|---|
| | | *Parameter I* | *Parameter II* | *Parameter III* | *Total* | *Average* |
| 200 | 10 | 379 | 18 | 10 | 407 | 135.67 |
| 400 | 10 | 798 | 21 | 14 | 833 | 277.67 |
| 600 | 10 | 1310 | 25 | 16 | 1351 | 450.33 |
| 800 | 10 | 1768 | 27 | 20 | 1815 | 605.00 |
| 1000 | 10 | 2502 | 29 | 21 | 2552 | 850.67 |
| 1200 | 10 | 3122 | 30 | 23 | 3175 | 1058.33 |
| 1400 | 10 | 3599 | 31 | 23 | 3653 | 1217.67 |
| 1600 | 10 | 4218 | 32 | 24 | 4274 | 1424.67 |

Table 3 presents three experimental results obtained for various scenarios using statistical methods. The time taken for each of the mathematical models, which is the assignment and correlation method, was found and then their total average time for which the best matched resource to a job was computed. However, it was observed that as the number of resources increase, the average time taken to find the best matched resource increases gradually.

**Table 3** Descriptive statistics

| | *Mean* | *Std. deviation* | *N* |
|---|---|---|---|
| number of resources | 900.00 | 489.898 | 8 |
| average response time (sec.) | 752.50 | 461.422 | 8 |

The average number of resources is 900. The mean of the average response time (sec.) is 752.5. According to Table 3, the standard deviation of the number of resources is 489.9 and the standard deviation for the average response time (sec.) is 461.4. Standard deviation here implies the measures of the spread or how far each of the set of resource is from the mean or centre of the available resource distribution.

Table 4 indicates that the value of the coefficient of determination is 0.99 which shows that the average response time (sec.) explain the variation of the number of resources around their mean value of 900 for about 99%.

**Table 4** Model summary

| *Model* | *R* | *R Square* | *Adjusted R Square* | *Std. Error of the Estimate* |
|---|---|---|---|---|
| 1 | .998[a] | .996 | .995 | 33.001 |

Notes: [a] Predictors: (Constant), average response time (sec.).

Table 5 shows that the explanatory power of the model is significant, that is, the 99% variation in the number of resources is accounted for by the average time (sec.) is significant. On the other hand only 1% of the variation in the number of resources is due to random happenings (residual).

**Table 5** ANOVA[b]

| | *Model* | *Sum of squares* | *df* | *Mean square* | *F* | *Sig.* |
|---|---|---|---|---|---|---|
| 1 | Regression | 1673465.693 | 1 | 1673465.693 | 1536.627 | .000[a] |
| | Residual | 6534.307 | 6 | 1089.051 | – | – |
| | Total | 1680000.000 | 7 | – | – | – |

Notes: [a] Predictors: (Constant), average response time (sec.).

[b] Dependent variable: number of resources.

According to Table 6, the value of the intercept term is 102.6, that is, if the average response time (sec.) is zero the number of average response time is expected to be 102.6. The slope of the model is 1.1 indicating that if the average response time (sec.) changes by one, averagely the number of resources changes by 1.1. The *p*-value of the coefficient being less than the theoretical level of significance which is 5%, we can conclude that the intercept term and the slope of the regression model are significant. This further shows that there is a significant relationship between the average response time (sec.) and the number of resources.

**Table 6** Coefficients[a]

| Model | | Unstandardised coefficients | | Standardised coefficients | t | Sig. | 95% confidence interval for $\beta_1$ | |
|---|---|---|---|---|---|---|---|---|
| | | $\beta_1$ | Std. Error | $\beta_1$ | | | Lower bound | Upper bound |
| 1 | (Constant) | 102.614 | 23.450 | – | 4.376 | .005 | 45.234 | 159.995 |
| | average response time (sec.) | 1.060 | .027 | .998 | 39.200 | .000 | .994 | 1.126 |

Notes: [a] Dependent variable: number of resources.

We explain below some of the terms in Table 6:

$$t = \frac{\hat{\beta}_i - \beta_i}{Se\left(\hat{\beta}_i\right)} \sim t_{n-2}^{\alpha/2}, \; Se\left(\hat{\beta}_2\right) = \sqrt{\sigma_\mu^2 \frac{1}{n\sum x_i^2}}, \; \hat{\sigma}_\mu^2 = \frac{\sum e_i^2}{n-2},$$

$$e_i = Y_i - \hat{Y}_i$$

$t$ = standardised value of $\hat{\beta}_i$

$t_{n-2}^{\alpha/2}$ = tabulated value of $t$-statistic

$x_i^2$ = square deviation of the number of response time from their mean

$e_i$ = the error term (observable)

$\hat{\sigma}_\mu^2$ = estimated variance of the stochastic disturbance term

$n$ = sample size

$\alpha$ = 5% (type I error)

Figure 9 shows the impact of scheduling users' job amidst large number of available grid resource. As the number of resources increase, the average time taken in seconds to find a suitable resources increases likewise, especially for the assignment technique which adopt a static means of matching jobs to resources. However, the remaining two models decrease the search time significantly irrespective of the number of jobs to resources ratio available at any given time.

**Figure 9** Measurement of time to allocate $n$ number of resources (see online version for colours)

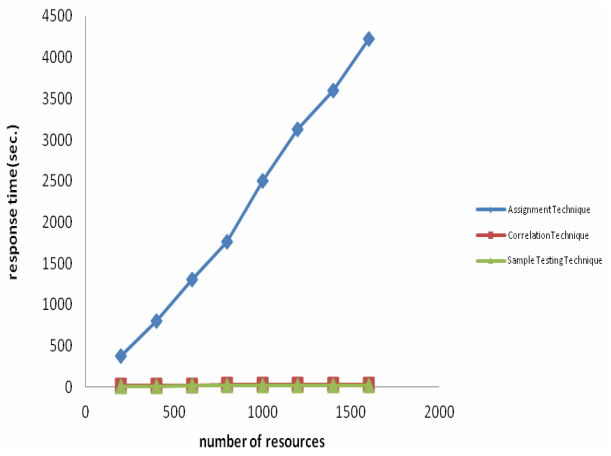
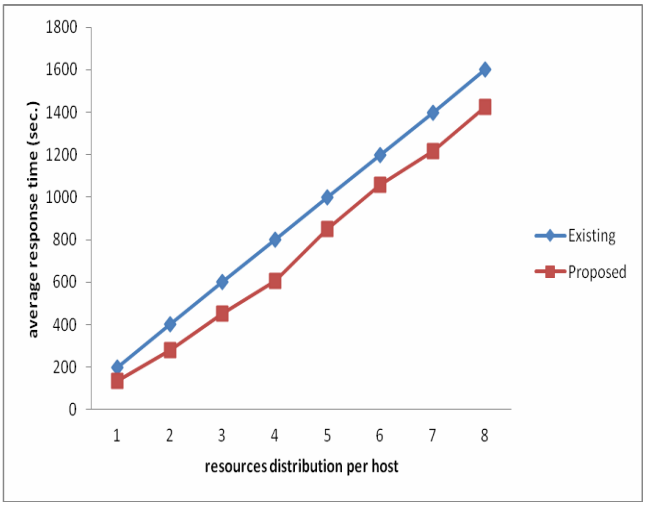


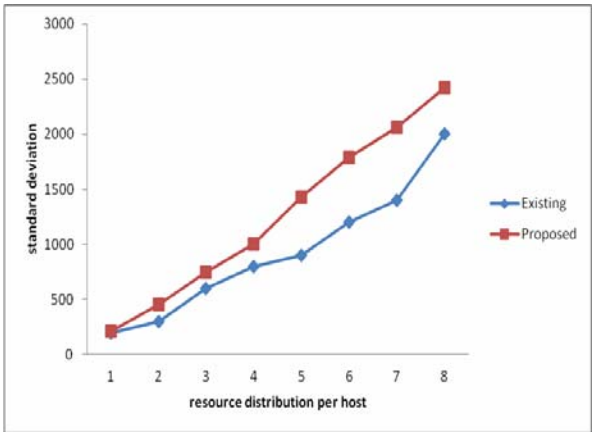

Figure 10 illustrates the average response time performance of the existing Condor-G matchmaker mechanism with our proposed model. One major disadvantage of the existing mechanism as stated earlier, is that the system simply assumes that it is dealing with serial jobs even when the jobs are distributed. However, the proposed job allocation mechanism is dynamic and has the capability of detecting job parallelism and likewise handles distributed job allocation efficiently.

**Figure 10** Comparison between the existing matchmaker algorithm with the proposed algorithm for the average response time taken to find the best matched resources (see online version for colours)




The measure of dispersion for both proposed and existing models based on the computed standard deviations is illustrated in Figure 11. The standard deviation indicates the measure of dispersion for the average response time with respect to resource distribution.

**Figure 11** Comparison of measure of dispersion for the average response time for the existing and proposed matchmaking methods (see online version for colours)



## 7 Conclusion and future directions

This paper addresses some of the challenges that could be encountered in the course of developing an efficient scheduling model for solving resource allocation problems in a heterogeneous distributed problem solving environment such as the scientific virtual laboratory framework. Resource management system and scheduling models are proposed for allocating resource to users' submitted jobs and scheduling of multiple workloads for scientific virtual laboratory applications. The experimental result obtained indicates that the proposed model will be efficient enough to perform multiple tasks scheduling with shareable and distributed grid resources. It is also anticipated that the proposed model should be able to balance the processing workloads of users' submitted jobs efficiently. We are planning to adapt the proposed model in dynamic real-time processing environments to collaborative scientific experiment application. With such improvements, the model can be integrated into the virtual laboratory scheduling scheme in order to improve their performance.

The proposed framework described in this paper is a step in the direction of providing a flexible environment to support both current and future applications and their emerging requirements for the challenging issues that come with various workflow scenarios in the virtual laboratory framework described by Handschuh et al. (2009), Lawenda et al. (2004b) and Imamagic et al. (2006).

## References


Afsarmanesh, H., Benabdelkader, A., Kaletas, E.C., Garita, C. and Hertzberger, L.O. (2000) 'Towards a multi-layer architecture for scientific virtual laboratories', *High Performance Computing and Networking*, Springer, Berlin Heidelberg, pp.163–176.

Afsarmanesh, H., Kaletas, E.C., Benabdelkader, A., Garita, C. and Hertzberger, L.O. (2001) 'A reference architecture for scientific virtual laboratories', *Future Generation Computer Systems*, Vol. 17, No. 8, pp.999–1008.

Bai, X., Yu, H., Ji, Y. and Marinescu, D.C. (2004) 'Resource matching and a matchmaking service for an intelligent grid', *International Conference on Computational Intelligence*, 17–19 December, Istanbul, Turkey, pp.262–265.

Berman, F., Casanova, H., Chien, A., Cooper, K., Dail, H., Dasgupta, A., Deng, W., Dongarra, J., Johnsson, L., Kennedy, K., Koelbel, C., Liu B., Liu, X., Mandal, A., Marin, G., Mazina, M., Mellor-Crummey, J., Mendes, C., Olugbile, A., Patel, M., Reed, D., Shi, Z., Sievert, O., Xia, H. and YarKhan, A. (2005) 'New grid scheduling and rescheduling methods in the GrADS project', *International Journal of Parallel Programming*, Vol. 33, No. 2, pp.209–229.

Berstis, V. (2002) 'Fundamentals of grid computing', *IBM Redbooks Paper*, pp.1–28.

Bhanu, S.M.S. and Gopalan, N.P. (2008) 'A hyper-heuristic approach for efficient resource scheduling in grid', *International Journal of Computers, Communication and Control*, Vol. 3, No. 3, pp.249–258.

Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I. and Freund, R.F. (2001) 'A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems', *Journal of Parallel and Distributed Computing*, Vol. 61, No. 6, pp.810–837.

Bu, Y-P., Wei, Z. and Yu, J-S. (2008) 'An improved PSO algorithm and its application to grid scheduling problem', *International Symposium on Computer Science and Computational Technology, ISCSCT'08*, 20–22 December, Shanghai, China, Vol. 1, pp.352–355.

Buyya, R. and Murshed, M. (2002) 'Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing', *Concurrency and Computation: Practice and Experience*, Vol. 14, Nos. 13–15, pp.1175–1220.

Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W. and Tuecke, S. (1998) 'A resource management architecture for metacomputing systems', *Job Scheduling Strategies for Parallel Processing*, Springer, Berlin Heidelberg, pp.62–82.

Czajkowski, K., Foster, I., Kesselman, C., Sander, V. and Tuecke, S. (2002) 'SNAP: a protocol for negotiating service level agreements and coordinating resource management in distributed systems', *Job Scheduling Strategies for Parallel Processing*, Springer, Berlin Heidelberg, pp.153–183.

Foster, I. and Kesselman, C. (Eds) (2003) *The Grid 2: Blueprint for a New Computing Infrastructure*, Access Online via Elsevier.

Foster, I., Kesselman, C. and Tuecke, S. (2001) 'The anatomy of the grid: enabling scalable virtual organizations', *International Journal of High Performance Computing Applications*, Vol. 15, No. 3, pp.200–222.

Frey, J., Tannenbaum, T., Livny, M., Foster, I. and Tuecke, S. (2002) 'Condor-G: a computation management agent for multi-institutional grids', *Cluster Computing*, Vol. 5, No. 3, pp.237–246.

Frincu, E.M. (2011) *Adaptive Scheduling for Distributed Systems*, Doctorial Dissertation, West University of Timisoara, Timisoara, Romania.

Galstyan, A., Czajkowski, K. and Lerman, K. (2005) 'Resource allocation in the grid with learning agents', *Journal of Grid Computing*, Vol. 3, Nos. 1/2, pp.91–100.

Gao, Y., Rong, H. and Huang, J.Z. (2005) 'Adaptive grid job scheduling with genetic algorithms', *Future Generation Computer Systems*, Vol. 21, No. 1, pp.151–161.

Gruber, T.R. (1993) 'A translation approach to portable ontology specifications', *Knowledge Acquisition*, Vol. 5, No. 2, pp.199–220.

Han, W., Shi, X. and Chen, R. (2008) 'Process-context aware matchmaking for web service composition', *Journal of Network and Computer Applications*, Vol. 31, No. 4, pp.559–576.

Handschuh, L. (2012) *Virtual Laboratories as the New Approach to Genomic Experiments*. Available online at: http:// lib. bioinfo. pl/courses/question_list/515 (accessed on 15 January 2012).

Handschuh, L., Lawenda, M., Stępniak, P., Figlerowicz, M., Stroiński, M. and Węglarz, J. (2009) 'New approach to genomics experiments taking advantage of virtual laboratory system', *Computational Methods in Science and Technology*, Vol. 15, No. 1, pp.31–40.

Imamagic, E., Radic, B. and Dobrenic, D. (2006) 'An approach to grid scheduling by using condor-G matchmaking mechanism', *Journal of Computing and Information Technology*, Vol. 14, No. 4, pp.329–336.

Izakian, H., Ladani, B.T., Zamanifar, K. and Abraham, A. (2009) 'A novel particle swarm optimization approach for grid job scheduling', *Information Systems, Technology and Management*, Springer, Berlin Heidelberg, pp.100–109.

Jacob, J., Rajsingh, B.E. and Jesudasan, I.B. (2011) 'Three dimensional matchmaking model for optimal allocation of resources in grid', *European Journal of Scientific Research*, Vol. 67, No. 1, pp.128–136.

Kuokka, D. and Harada, L. (1995) 'Matchmaking for information agents', *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI'95*, 20–25 August, Montreal, Quebec, Canada, Vol. 1, pp.672–678.

Lawenda, M., Meyer, N., Rajtar, T., Okon, M., Stokłosa, D., Kaliszan, D. and Stroiński, M. (2004a) 'Generalization aspects in the virtual laboratory system', *Poznan Supercomputer and Networking System*. Available online at: http://www.psnc.pl/ (accessed on 07 January 2012).

Lawenda, M., Meyer, N., Rajtar, T., Okon, M., Stoklosa, D. and Stroinski, M. (2004b) 'Job workflow in the virtual laboratory', *Global Grid Forum*, Vol. 10, pp.1–9.

Liu, S. (2007) 'User-centric resource allocation hierarchy in grid computing', *6th International Conference on Grid and Cooperative Computing, GCC'07*, 16–18 August, Los Alamitos, CA, USA, pp.228–235.

Liu, Y. and He, H. (2007) 'Grid resource discovery approach based on matchmaking engine overlay', *3rd International Conference on Semantics, Knowledge and Grid*, 29–31 October, Shan Xi, China, pp.294–297

Moreno, R. and Alonso-Conde, A.B. (2004) 'Job scheduling and resource management techniques in economic grid environments', *Grid Computing*, Springer, Berlin Heidelberg, pp.25–32.

Ramakrishan, L., Nurmi, D., Mandal, A., Koelbel, C., Gannon, D., Huang, M., Kee, Y-S., Obertelli. G., Thyagaraja, K., Wolski, R., YarKhan, A. and Zagorodnov, D. (2009) 'VGrADS: enabling e-Science workflows on grids and clouds with fault tolerance', *High Proceedings of the Conference on Performance Computing Networking, Storage and Analysis*, 14–20 November, Portland, Oregon, pp.1–12.

Ritchie, G. and Levine, J. (2004) 'A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments', *Proceedings of the 23rd Workshop of the UK Planning and Scheduling Special Interest Group*, Cork.

Shojafar, M., Barzegar, S. and Meybodi, M.R. (2010) 'A new method on resource scheduling in grid systems based on hierarchical stochastic Petri net', *Proceedings of 3rd International Conference on Computer and Electrical Engineering (ICCEE 2010)*, 16–18 November, Chengdu, Sichuan, China, pp.175–180.

Shu, G., Rana, O.F., Avis, N.J. and Dingfang, C. (2007) 'Ontology-based semantic matchmaking approach', *Advances in Engineering Software*, Vol. 38, No. 1, pp.59–67.

Thain, D., Tannenbaum, T. and Livny, M. (2005) 'Distributed computing in practice: the condor experience', *Concurrency and Computation: Practice and Experience*, Vol. 17, Nos. 2–4, pp.323–356.

Venugopal, S. (2006) *Scheduling Distributed Data-Intensive Applications on Global Grids*, PhD Thesis, The University of Melbourne, Melbourne, Australia, pp.82–83.

Weitzel, D. (2011) *Campus Grids: A Framework to Facilitate Resource Sharing*, Doctoral Dissertation, University of Nebraska, Lincoln, Nebraska, USA.

## Notes

1 Virtual Laboratory is generally perceived as a heterogeneous, distributed environment, which allows scientists and engineers from different geographical location to conduct experiments with the usage of physical laboratory devices, perform simulation using computational application software; enable communication and collaboration among users working on the same group of research projects.

2 From service connection point of view, VL can be divided into three levels of operational activities: the user interface, the application server and the device server. More so, from architectural point of view it can be portioned into four layers: Access layer, Grid layer, Supervisory layer and Resources layer. Also see Lawenda et al. (2004b).

3 Conceptualisation is an abstract and simplified view of the world that we wish to represent for some purpose.