International Journal of Innovative Research in Engineering & Management (IJIREM)
ISSN: 2350-0557, Volume-2, Issue-3, May-2015

DYNAMIC INTELLIGENT DUAL QUEUE MATCHMAKING MODEL (DIDQMM) FOR RESOURCE ALLOCATION IN GRID ENVIRONMENT

Dr. J. Japhynth
Professor, Dr.G.U.Pope College of Engg.,
TamilNadu, India.

Dr.J.R.Isaac Balasingh
Dean, Dr.G.U.Pope College of Engg.,
TamilNadu, India.

ABSTRACT

Grid resources can be distributed in different geographical regions. Matching jobs with distributed resources is one of the key issues in grid computing. In traditional matchmaking algorithms, matchmaking is based on static attributes; the dynamicity of the resource pool is not considered; the final ranks are not normalized and there is no fairness in allocation of jobs. This leads to incomplete jobs, job failures, user dissatisfaction and infinite waiting of jobs. Hence selecting appropriate resource for a job plays a vital role in resource allocation. The resource matching mechanism must consider the dynamically changing conditions both in terms of resource demand and resource availability. The request submitted by the user may have different constraints that can only be satisfied by certain types of resources with specific capabilities. Resource matching mechanisms must be able to match multiple jobs to a single resource, provided the resource has enough capacity to handle the matched jobs simultaneously. Resource matching mechanism should also consider multiple parameters in matchmaking process to select the best available resources such that the capacity consumed by the matched jobs does not exceed the total resource capacity. Therefore a novel Dynamic Intelligent Dual Queue Matchmaking Model *(DIDQMM)* is proposed, which considers static, dynamic and behavioral parameters and Dual Queues to allocate best resource to the job.

Keywords

Matchmaking, parameters, allocation, static, dynamic, Intelligent.

1. INTRODUCTION

Matching is a common operation in many areas of computer science.. In a Grid environment, the resource allocation process must select resources appropriate to the requirements of the application. This process of selecting resources based on application requirements is called "resource matching". In computer science, the term *matching* refers to the process of evaluation of the degree of similarity of two objects. Nowadays many matchmaking processes address the issue of job- resource matching. The job-resource matchmaking process involves three types of agents: consumers (*requesters)*, producers (*providers)*, and a matchmaking service. A matchmaking service uses the matching algorithm to evaluate a matching function which computes the matching degree. Users describe their applications with Classads language and submit them to Matchmaker. On the other side, resources publish information to the Matchmaker through Classads. Classads allows defining custom attributes for resources and jobs. Matchmaker then matches job requests with available resources. In matchmaking, each object is characterized by a set of properties/attributes; each property is a tuple (*name*, *value*), with a *name,* a string of characters and *value* either a constant (a number – integer or real, a Boolean constant – "true" or "false", or a string of characters), or an expression that returns a constant. The "matching degree", *m*, is a real number. Typically, $0< m< 1$, with $m=0$ is a *total mismatch* and $m=1$ is a *perfect match.* Matching shared resources with the request in the grid raises many challenges such as large pool of resources; heterogeneity of platforms; the diversity in user behavior; dynamic nature of the resources in the resource pool and lack of reliability.

2. THREE DIMENSIONAL PARAMETERS

The three dimensional parameters are static, dynamic and behavioral. The user provides the static parameters and its expected dynamic parameters and behavioral parameters. Similarly the service provider provides the static and dynamic parameters for each of its resources. The behavioral parameter of each resource of the service provider is determined by the knowledge unit of the Dynamic Intelligent Dual Queue Matchmaking Model *(DIDQMM).* The three dimensional parameters and their notations are given in table 2.1.

Table 2.1 Parameter names and their notations

Dimensions	Parameter name	Notations
1st dimension (Static)	Resource type	R_r
	Resource name	R_n
	Resource location	R_l
	Resource CPU type	R_c
	Resource OS	R_o
	Memory capacity	R_m
2nd Dimension (Dynamic)	Hard disk availability	D_a
	Economic state	E_s
	Dedication rate	D_r
	Resource cost	R_{co}
	Deadline	D_l
3rd Dimension (Behavioral)	Success rate	S_r
	Hit rate	H_r
	Failure rate	F_r
	Recovery time	R_t

3. PROPOSED NOVEL DYNAMIC INTELLIGENT DUAL QUEUE MATCHMAKING MODEL (DIDQMM)

An important issue in a grid is the allocation of jobs to the best resources. The proposal aims at allocating optimal resource for a job using various components of the *DIDQMM*. Traditional matchmaking model has a single queue of jobs which are ready to be processed. Jobs may experience delays inside the single queue which results in increase in the average response time. The job whose resource is readily available needs to wait in the queue until the previous jobs are served. This delays the job even though its resources are available. This motivates the researcher to design the Dynamic Intelligent Dual Queue Matchmaking Model *(DIDQMM)* which maintains two queues [16] in the resource analyst. The proposed Novel Dynamic Intelligent Dual Queue Matchmaking Model *(DIDQMM) is* shown in Fig.3.1 and has two main components. They are Grid Manager and Resource Monitor. The functions of each component in the Dynamic Intelligent Dual Queue Matchmaking Model are explained in the following sections.

3.1 Grid Manager

Grid manager is responsible for managing all grid activities. It is an interface between the user and the service provider. It has Job handler, Resource handler, 3D Matchmaking engine and the Accounting facilitator. They are explained in this section.

Job handler

The functions of the job handler are: User registration, Job submission handling, Job table maintenance, Job classification, Job status checking and optimal payment intimation. The job handler stores the job's descriptions along with the *User_name*, *user_id*, *Job_name*, *Job_id* and the arrival time of the job in the job table.

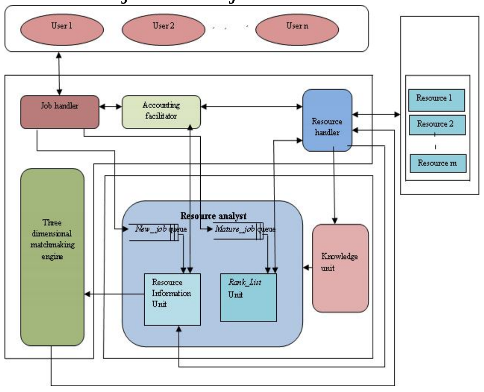


Fig 3.1 Dynamic Intelligent Dual Queue Matchmaking Model *(DIDQMM)*

DIDQMM calculates the *rank_list* (list of 5 crest matched resources) for every submitted job. After the user submits the job to the job handler, the job handler verifies the job table to know whether the submitted job has been previously processed, by calculating the similarity Index *SI* [15] of the submitted job with all jobs in the job table whose time of arrival is less than ∆t (where ∆t is the time interval at which the resource's dynamic and behavioral parameters are updated periodically by the service providers and the knowledge unit respectively). The similarity index *SI* is calculated using the Eq. 3.1.

$$S = \frac{2n_{xy}}{(n_x + n_y)} \qquad (3.1)$$

Where n_{xy} is the number of parameters matched with the job in the job table, n_x is the number of parameters of the job in the Job table and n_y is the number of parameters of the submitted job. If the submitted job's similarity index *SI* is equal to 1 with respect to a particular job in the job table, the job handler identifies the submitted job as "*mature* (m_j)". The job handler then forwards the matched *job_id* along with the submitted job to the resource analyst. The matched *job_id* is sent along with the submitted job in order to get the matched job's already calculated *rank_list* from the *Rank_list* table in the resource analyst. The submitted job is then forwarded to the resource handler for allocation of resource. If the submitted job's similarity index *SI* is not equal to 1 with respect to a particular job in the job table, the job handler identifies the job as "*new*

(n_j)". The submitted job is then forwarded to the 3D matchmaking engine for rank calculation.

Resource handler

The functions of the resource handler are: Service provider registration, Resource registration, Resource table maintenance, Resource rescheduling, Behavioral parameters (Success rate (S_r) and recovery time(R_t)) updation and Job completion intimation. The resource handler maintains a resource table. For each submitted job, a *rank_list* – a list of 5 crest matched resources is calculated by the 3D matchmaking engine and forwarded to the resource analyst and to the resource handler. The resource handler stores the *rank_list* in the resource table. The *rank_list* is stored in the Rank_list table of the resource analyst to be used by the "*mature* (m_j)" jobs. The *rank_list* is stored in the resource table for *rescheduling* of resources in case of resource/job failures. Rescheduling is defined as the redispatching of resources to the job from the rank list. The resource handler performs rescheduling of resources by dispatching the next highly ranked resource in the *rank_list* to the job. The rescheduling occurs in the following cases. Case 1: If the user voluntarily quits the resource. Case 2: If the service provider quits the job in case of arrival of higher priority job.

Accounting facilitator

The functions of the accounting facilitator are: optimal payment [13] calculation and accounting table maintenance. The optimal payment is calculated using the Eq. 3.2.

$$U_i^j = \frac{(er_i^n\ q_{in}\ p_j)^{\frac{1}{2}}}{c_j}\ \frac{\sum_{k=1}^{n}\left(\frac{q_{ik}\ p_k}{er_i^n\ c_j}\right)^{\frac{1}{2}}}{T_i} \qquad (3.2) \qquad (3.2)$$

where U_i^j is the optimal payment of grid user *i* to service provider *j* under completion time constraint to maximize the user's benefits. x_i^j is the resource allocated to grid user *i* by grid service provider *j*. c_j is capacity of the resource *j*, er_i^n is the energy consumption rate, q_{in} is the size of the *n* th job, p_j is the price of the unit resource in the service provider *j*, T_i is the time limits given by the *i* th grid user to complete the job.

Table 3.2 Resource table

User_id	*Job_id*	*Resource _id*	*Rank*	Job status	Resource status
5	8	5	1	INEXECUTION	INEXECUTION
5	8	12	2		RANKED
5	8	27	3		RANKED
5	8	67	4		RANKED
5	8	68	5		RANKED
8	1	6	1	INEXECUTION	CANCELLED
8	1	4	2		CANCELLED
8	1	9	3		INEXECUTION
8	1	13	4		RANKED
8	1	3	5		RANKED

3D Matchmaking engine

This is the core part of the proposed Dynamic Intelligent Dual Queue Matchmaking Model *(DIDQMM)*. The primary function of the 3D matchmaking engine is to match the user requirements of the submitted job with the resource capabilities. The resource analyst performs the preliminary selection of resources using the static parameter viz. resource type (R_r) and forwards the selected resource's capabilities (static, dynamic and behavioral parameter values) and the submitted job with user requirements to the 3D matchmaking engine. The functions of the 3D matchmaking engine are: Static parameters matching using Assignment method, Dynamic and behavioral parameters matching using correlation method, *Rank_list* calculation, and Behavioral parameters (Hit rate (H_r) and failure rate(F_r)) updation.

The 3D Matchmaking engine uses the Assignment method to match the job's static parameter values with the resource's static parameter values. Assignment method is used for the optimal assignment of a static parameter to a resource to ensure maximum efficiency. This method wraps up the assignment of a static parameter to a resource. The static parameters are Resource name (R_n), Resource location (R_l), Resource CPU type (R_c), Resource OS (R_o) and Memory capacity (R_m). Let X_{ik} represents job parameter of i-th user where k varies from 1 to n where n is the number of parameters, R_s represents resources where s varies from 1 to m where m is the number of resources and R_{sk} represents resource parameters where s varies from 1 to m and k varies from 1to n where n and m are number of parameters and number of resources respectively.

For each submitted job, the 3D matchmaking engine computes a relative parameter match value (C_{isk}) by comparing the job's static parameters and the preliminarily selected resource's static parameters. For X_{ik} where k represents the static parameter R_n and R_l, C_{isk} is calculated using Eq. 3.3.

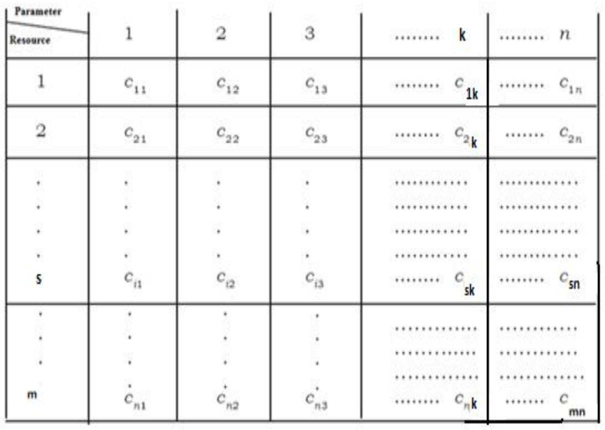
$$C_{isk} = \begin{cases} 1 & if\ X_{ik} = R_{sk} \\ X_{ik} - R_{sk} & otherwise \end{cases} \quad (3.3)$$

For X_{ik} where k represents the static parameter R_c, $R_o$and R_m , C_{sk} is calculated using Eq. 3.4.

$$C_{isk} = \begin{cases} 1 & if\ X_{ik} \le R_{sk} \\ X_{ik} - R_{sk} & if\ X_{ik} > R_{sk} \end{cases} \quad (3.4)$$

The relative parameter match values of R_1 compared with X_{ik} where k varies from 1 to n are $C_{11,}\ C_{12\,,} \dots , C_{1k\,,} \ {}_{\dots,}C_{1n}$
The relative parameter match values of R_2 compared with X_{ik} where k varies from 1 to n are $C_{21,}\ C_{22\,,}, C_{2k\,,}\ {}_{\dots,}C_{2n}$
The relative parameter match values of R_m compared with X_{ik} where k varies from 1 to n are $C_{m1,}\ C_{m2\,,} \dots , C_{mk\,,}\ {}_{\dots,}C_{mn}$
Let there be n number of job parameters in a job and m number of resources available for matchmaking with different parameter values. If the relative match value of assigning k_{th} parameter of the i-th user's job to s_{th} resource is c_{isk}, the assignment is shown in Table 3.3.

Table 3.3 Assignment table



Parameter / Resource	1	2	3	.... k	.... n
1	c_{11}	c_{12}	c_{13}	.... c_{1k}	.... c_{1n}
2	c_{21}	c_{22}	c_{23}	.... c_{2k}	.... c_{2n}
.	.	.	.	....	....
.	.	.	.	....	....
.	.	.	.	....	....
.	.	.	.	....	....
s	c_{i1}	c_{i2}	c_{i3}	.... c_{sk}	.... c_{sn}
.	.	.	.	....	....
.	.	.	.	....	....
.	.	.	.	....	....
m	c_{n1}	c_{n2}	c_{n3}	.... c_{nk}	.... c_{mn}

Table 3.4 Weight value of static parameters

S. No	Static parameter	Weight value
1	Resource name R_n	0.4
2	Resource location R_l	0.05
3	Resource CPU type R_c	0.1
4	Resource OS R_o	0.15
5	Memory capacity R_m	0.3

The 3D matchmaking engine uses the correlation method to match the job's dynamic and behavioral parameter values with the resource's dynamic and behavioral parameter values. Correlation method is used to find the correlation between the dynamic and behavioral parameters of the submitted job and the resource. Correlations follow two patterns. They are positive correlation and negative correlation. In a positive correlation, as the values of one of the variables increase, the values of the second variable also increase. In a negative correlation, as the values of one of the variables increase, the values of the second variable decrease. Positive and negative correlations range in their strength from weak to strong. A zero correlation occurs when there is no relationship between variables. Positive correlations will be reported as a number between 0 and 1. A score of 0 indicates that, there is no correlation (the weakest measure). A score of 1 is a perfect positive correlation. As the correlation score gets closer to 1, it is getting stronger. So, a correlation of .8 is stronger than .6; but .6 is stronger than .3. The negative sign does not indicate anything about strength. It indicates that the correlation is negative in direction. While judging the strength of a correlation, the sign is ignored. The dynamic parameters are Hard disk availability (D_a), Economic state (E_s), Dedication rate (D_r), Resource cost (R_{co}) and Deadline (D_l). The behavioral parameters are Success rate (S_r), Hit rate (H_r), Failure rate (F_r) and Recovery time (R_t). Let X_{ik} represents i-th user's job parameter where k varies from 1 to n where n is the number of parameters, R_s represents resources where s varies from 1 to m , R_{sk} represents resource parameters where k varies from 1to n where n and m are number of parameters and number of resources respectively.
The resource parameter values for R_s where s varies from 1 to m are as follows

$$R_{1k} = \{R_{11,}R_{12}, R_{13,}R_{14}, \dots , R_{1n}\}$$
$$R_{2k} = \{R_{21,}R_{22}, R_{23,}R_{24}, \dots , R_{2n}\}$$
$$R_{mk} = \{R_{m1,}R_{m2}, R_{m3,}R_{m4}, \dots , R_{mn}\}$$

Where k varies from 1 to n.
The correlation of any resource $R_s$with the job X_i of the i-th user is given in Eq. 3.5.

$$\boldsymbol{R_s} = \frac{k.\sum_{k=1}^{n} X_{ik}R_{sk} - \sum_{k=1}^{n} X_{ik} \cdot \sum_{k=1}^{n} X_{ik}R_{sk}}{\sqrt{(k.\sum_{k=1}^{n}(X_{ik})^2 - \left(\sum_{k=1}^{n} X_{ik}\right)^2) * (k.\sum_{k=1}^{n}(R_{sk})^2 - (\sum_{k=1}^{n} R_{sk})^2)}}$$
(3.5)

Where n is number of dynamic and behavioral parameters. The resources are ranked according to the correlated value of the resource parameter with the job parameter. The resource whose correlated value is nearly equal to one is given the highest rank.

3.2 Resource Monitor

The resource monitor monitors the resources in the grid site through the resource handler. It contains two components. They are Resource analyst and Knowledge unit.

Resource analyst

Resource analyst is the central part of the matchmaking architecture. It maintains two queues viz. *New_job* queue and *Mature_job* queue. *New* jobs are queued in the *new_job* queue and *mature* jobs are queued in the *mature_job.* When the user submits the job to the job handler, the job handler identifies the job as "*new*(n_j)" or "*mature*(m_j)" and forwards the job to the resource analyst. When the job handler identifies the job as "*new*(n_j)", it forwards the "*new*(n_j)" job with user requirements to the resource analyst. When the job handler identifies the job as "*mature*(m_j)", it forwards the "*mature*(m_j)" job and the matched *job_id* to the resource analyst. The resource analyst queues the "*new*(n_j)" job in the *new_job queue* and the "*mature*(m_j)" job in the *mature_ job* queue. The resource analyst comprises of two components viz. Resource Information Unit (RIU) and *Rank_List* Unit (RLU). The interactions between the components in the resource analyst are shown in Fig 3.2.

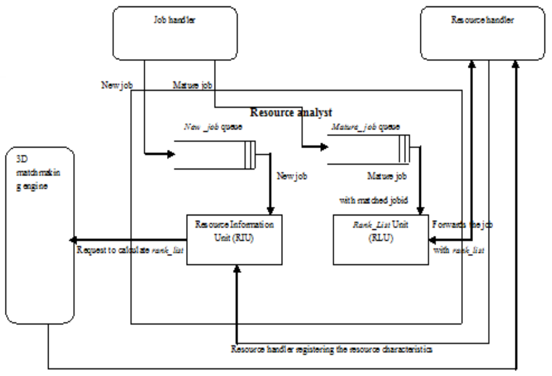

Job handler
Resource handler
New job
Mature job
Resource analyst
New _job queue
Mature_job queue
3D matchmaking engine
New job
Mature job with matched jobid
Resource Information Unit (RIU)
Rank_List Unit (RLU)
Forwards the job with rank_list
Request to calculate rank_list
Resource handler registering the resource characteristics

Fig 3.2 Interactions between the components in the Resource analyst

The Resource Information Unit (RIU) receives the "new(n_j)" job with the user requirements from the job handler and the *Rank_List* Unit (RLU) receives the "*mature*(m_j)" job with the matched *Job_id* from the job handler. The Resource Information Unit maintains Resource Information Table and the *Rank_List* Unit maintains *Rank_List* Table.

Resource Information Unit (RIU) performs the preliminary resource selection by considering the static parameter (i.e., Resource type (R_r)). When the user submits "*new* (n_j)" job, the resource analyst performs the preliminary selection of resources by considering the first static parameter (i.e., Resource type (R_r)). The resource analyst selects all the resources of that particular resource type. The resource analyst then forwards the three dimensional parameter values of the selected resources and the submitted job along with the user requirements to the 3D Matchmaking Engine for *rank_list* calculation. For *mature* jobs, the *Rank_List* unit (RLU) fetches the *rank_list* of the matched *job_id* from the *rank_List* table and forwards the "*mature*(m_j)" job with the *rank_list* to the resource handler for allocation.

Knowledge unit

The knowledge unit receives the information about the history of behavior of resources in the resource pool from the 3D matchmaking engine and the resource handler. It receives the 'U' and 'Q' values of resources for success rate calculation and recovery time profiles of resources for recovery time calculation from the resource handler. It receives the hit rate of a resource at every Δt_1 time and 'R' and 'P' values of resources for failure rate calculation from the 3D matchmaking engine. The knowledge unit predicts the success rate, hit rate, failure rate and recovery time for each resource for the next matchmaking cycle based upon its previous behavior with the jobs. It also updates the predicted values of success rate, hit rate, failure rate and recovery time for each resource in the resource information table of resource analyst at every Δt time interval. It maintains the behavior parameter table (Table 5.7) and stores the predicted success rate (S_r), Hit rate (H_r), Failure rate (F_r) and Recovery time(R_t) of the resources in it. The behavior parameter table is shown in Table 3.7.

Table 3.7 Behavior parameter table

Resource_id	S_r	H_r	F_r	R_t in s
R4	.9	.7	.5	200
R6	.7	.6	.6	0
R8	.5	.5	.9	1000
R9	.7	.8	.4	500
R17	.4	.6	.3	300

4. PERFORMANCE EVALUATION

EXPERIMENTAL SETUP

This section provides the experimental evaluation of the performance of the *DIDQMM*. The DIDQMM is compared with our previous work, Three Dimensional Matchmaking Model (*TDMM)* [12]. The vital performance metrics have been identified to evaluate the proposed novel *DIDQMM.* The performance metrics viz. Average number of successful jobs, Average number of failed jobs and the average response time has been evaluated for various user and resource population profiles. The performance is analyzed for various scenarios. The description of the vital performance metrics considered for the evaluation of the *DIDQMM* is as follows:

Average number of successful jobs: It is the average number of jobs that has been completed successfully.

Average number of failed jobs: It is the average number of jobs that cannot be completed successfully after all rescheduling.

Average response time: It is the average amount of time a user waits before the job has been granted a resource.

The *DIDQMM* was simulated using Gridsim with discrete event simulation in Java. The *DIDQMM*'s matchmaking process is represented as a sequence of events. Each event occurs at an instant of time and marks a change in the state of the Grid system. The grid workload comprises single-processor jobs, which are sent to the grid in batches. A batch submission is the set of jobs ordered at the time of arrival into the *DIDQMM.* New allocation procedures are integrated into *DIDQMM* by extending the AllocPolicy class. The simulated environment encompasses the DAS-3 grids, for a total of 5 grid sites and over 225 processors. Each sites of the combined system receives independent stream of jobs. The experiments were conducted with 75 service providers with varying number of resources. The experimental results are the average of the 10 sets of synthetic job streams for each load levels viz. 20%, 30%, 40%, 50%, and 60%.

Table 4.1 shows the simulation parameters.

Table 4.1 Simulation Parameters

Sl.No	Simulation parameters	Values
1	Number of Grid sites	2 to 5
2	Number of service providers	65 - 120
3	Simulation time	≅ 10,000 ms
4	Number of users	3000
5	System load	20% (min) to 60% (max)

Experiment 1:

Experiment is conducted to compare the number of jobs successfully completed by the *CONDOR [6], FCONDOR [6], TDMM [12]* and *DIDQMM* under different input load levels. The experiment is conducted with 75 service providers evenly distributed in all grid sites. The number of jobs submitted ranges from 900 to 1800. The simulation time is set to 50000ms. The experiment was started with minimal input load and then increased gradually. From the result we infer that the *DIDQMM* outperforms by successfully completing more number of jobs than *CONDOR, FCONDOR,* and *TDMM.*

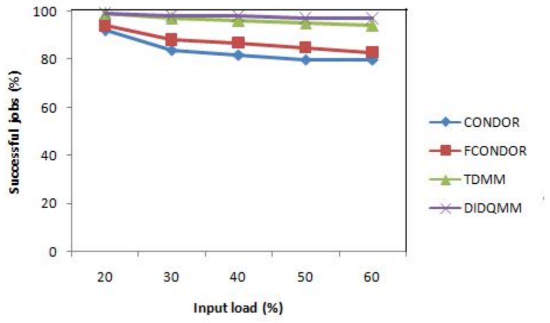


Fig. 4.1Successful jobs (%) Vs Input load (%) for CONDOR, FCONDOR, TDMM and DIDQMM

The Figure 4.1 shows the number of jobs completed by *CONDOR, FCONDOR, TDMM* and *DIDQMM.* It has been observed that the number of jobs completed by *DIDQMM* is relatively higher than the number of jobs completed by *CONDOR, FCONDOR* and *TDMM*. This is due to the fact that the dual queue in the resource analyst reduces the average waiting time of the job in the queue which provides room to successfully complete more number of jobs within the guaranteed time.

Experiment 2:

Experiment is conducted to compare the average response time of *TDMM* with *DIDQMM*. The experiment is conducted with 75 service providers evenly distributed in all grid sites. The number of jobs submitted ranges from 900 to 1800.

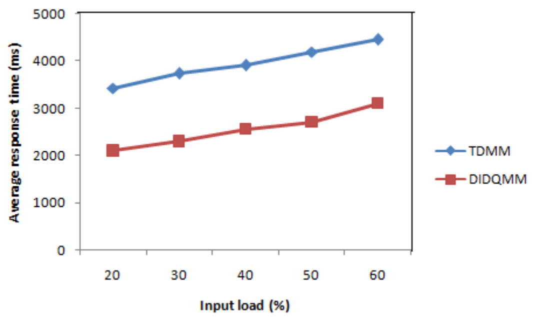


Fig 4.2 Average response time (s) Vs Input load (%) for *DIDQMM* and *TDMM*

The Figure 4.2 shows the average response time of *DIDQMM* and *TDMM.* It has been observed that the average response time of *DIDQMM* is relatively lower than the average response time of *TDMM*. This is due to the fact that the dual queue in the resource analyst reduces the average waiting time of the job in the queue which in turn reduces the average response time.

Experiment 3:
The experiment is conducted to compare the average response time with single dimensional parameters, single dimensional parameters with prior knowledge and three dimensional parameters with prior knowledge and Dual queue. The experiment is conducted with 75 service providers evenly distributed in all grid sites. The number of jobs submitted ranges from 900 to 1800.

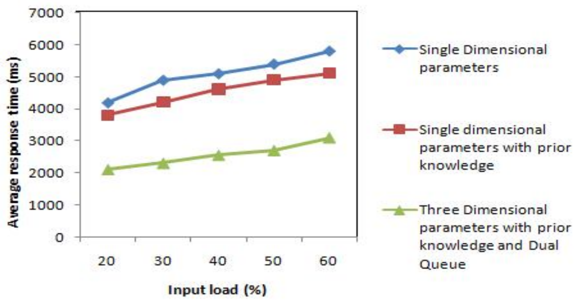


Fig 4.3 Comparative analysis of average response time (s) for single dimensional parameters, single dimensional parameters with prior knowledge and three dimensional parameters with prior knowledge and Dual queue for various input loads.

Figure 4.3 shows the Comparative analysis of average response time for single dimensional parameters, single dimensional parameters with prior knowledge and three dimensional parameters with prior knowledge and Dual queue for various input loads. From the graph it is inferred that the average response time of single dimensional parameters is higher than the single dimensional parameters with prior knowledge. This is due to the fact that the system with prior knowledge identifies the job as "*mature*(m_j)" and "*new*(n_j)" and uses the already calculated *rank_list* for "*mature*(m_j)" jobs. This reduces the average response time of a job. The figure 6.3 also infers that the system with three dimensional parameters, prior knowledge and dual queue shows low average response time than system with single dimensional parameters with prior knowledge. This is due to the fact that the dual queue in the resource analyst considerably reduces the waiting time of the job in the queue, which in turn reduces the average response time for job.

5. CONCLUSION

This paper provides the description of the proposed novel Dynamic Intelligent Dual Queue Matchmaking Model *(DIDQMM)*. The functions of various components of *DIDQMM* are presented. The proposed matchmaking model is designed, taking into consideration the successful completion of jobs, the job completion time, job migration and resource migration. The *DIDQMM* considers three dimension's parameters viz. static, dynamic and behavioral to select suitable resource for a job. This increases the precision of matchmaking and **best resource is selected for a job**. As the behavioral parameters of resources are considered in the matchmaking process, **the job's successful allocation is assured**. The updation of dynamic and behavioral parameters of the resources in the resource repository at every t time by the service providers and the knowledge unit respectively gives the up to date information to the 3D matchmaking engine for *rank_ist* calculation. Thus **the *TDMM* is reliable** and avoids ambiguity while preparing *rank_list* of resources. The performance is further improved by incorporating dual queue in the resource analyst *(DIDQMM)* viz. *mature_job queue* and *new_job queue.* The performance of *DIDQMM* (Dynamic Intelligent Dual Queue Matchmaking Model) was compared with *TDMM* (Three Dimensional Matchmaking Model). The average response time and the average number of successful jobs were obtained for *DIDQMM*. From the results, it has been found that the *DIDQMM* has minimized average response time than *TDMM* and increased average number of successful jobs than *TDMM*. Thus the proposed *DIDQMM* further minimizes the average response time and increases the average number of successful jobs. Hence it is inferred that *DIDQMM* outperforms *TDMM.*

6. FUTURE SCOPE

The following research directions can be focused in future to further enhance the system performance to a great extent
- Priorities can be set for jobs.
- User behaviors are studied, analyzed and included as one another dimension in matchmaking process.
- The user's matching threshold value can be obtained for each job and the matching may be done accordingly.

REFERENCES

[1] Amos Brocco, Apostolos Malatras, Beat Hirsbrunner , "Proactive Information caching for efficient resource discovery in self structured grid", *BADS'09*, June 19, 2009, Barcelona, Spain, Copyright 2009 ACM 978-1-60558-584-0/09/06.
[2] Anthony Sulistio, "Advance Reservation and Revenue-based Resource Management for Grid Systems" a thesis submitted in the Department of Computer Science and Software Engineering, The University of Melbourne, Australia.
[3] Chien-Min Wanga, Hsi-Min Chenb, Chun-Chen Hsuc, Jonathan Lee, "Dynamic resource selection heuristics for a non-reserved bidding-based Grid Environment", *Future Generation Computer Systems*, Vol. 26, pp. 183-197, 2010.
[4] Clematis A, A. Corana, D. D'Agostino, A. Galizia, A. Quarati, "Job resource matchmaking on Grid through two-level benchmarking", *Future Generation Computer Systems*, Vol. 26, Issue 8, pp. 1165-1179, Oct 2010.
[5] Daniel C. Vanderster, Nikitas J. Dimopoulosb, Rafael Parra-Hernandez, Randall J. Sobie, "Resource

allocation on computational grids using a utility model and the knapsack problem", *Future Generation Computer Systems*, Vol. 25, pp. 35-50, 2009.

[6] Emir Imamagic, Branimir Radic, Dobrisa Dobrenic, "An Approach to Grid Scheduling by using CondorG Matchmaking Mechanism", *Journal of Computing and Information Technology* – CIT, pp. 329–336, 2006.

[7] Eunjoung E tal, "Scheduling scheme based on dedication rate in volunteer computing environment", *in the proceedings of the 4th International Symposium on Parallel and Distributed Computing,* IEEE Computer Society Washington, DC, USA ©2005.

[8] Gao Shu, Omer F. Rana, Nick J. Avis, Chen Dingfang, "Ontology-based semantic matchmaking approach", *Advances in Engineering Software,* Vol. 38, pp. 59-67,2007.

[9] Guanfeng Liu, "Reputation Evaluation Framework based on Qos in Grid Economy Environments" *Quantitative Quality of service*, IGI global pages 219-232.

[10] Gustavo Sousa Pavani, Helio Waldman, "Co-scheduling in Lambda Grid Systems by means of Ant Colony Optimization", *Future Generation Computer Systems,* Vol. 25, pp. 257-265, 2009.

[11] Han Yu, Xin Bai, Dan C. Marinescu, "Workflow management and resource Discovery for an intelligent grid", *Parallel Computing,* Vol. 31, pp. 797-811, 2005.

[12] Japhynth Jacob, Elijah Blessing Rajsingh, Isaac Balasingh Jesudason "Three Dimensional Matchmaking Model for optimal Allocation of Resources in Grid" *European Journal of Scientific Research,* Vol 67, Issue 1, pp. 128-136.ISSN: 1450-216X/1450-202X, Dec 2011.

[13] Li Chunlin, Li Layuan, "Utility-based Scheduling for Grid Computing under Constraints of Energy Budget and Deadline" *Computer Standards & Interfaces,* doi: 10.1016/j.csi.2008.12.004, 2008.

[14] Martin KOLLÁR "Evaluation of real call set up success rate in GSM", *Acta Electrotechnical et Informatica,* Vol. 8, No. 3, pp.53-56, 2008.

[15] Michael Linch, "The similarity index and DNA fingerprinting", *Molecular Biological Evolution*, Vol. 7, pp 478-484, 1990.

[16] Rajiv Ranjan, Aaron Harwood and Rajkumar Buyya, "Case for Cooperative and incentive Based Federation of Distributed Clusters", *Future Generation Computer Systems,* Vol.24, pp. 280-295, May 2007.

[17] Rajkumar Buyya, Sudharshan Vazhkudai, "Compute Power Market: Towards a Market-Oriented Grid", *in the proceedings of the First IEEE International Symposium on Cluster Computing and the Grid (CCGrid'01),* pp.574, ISBN: 0-7695-1010-8, 2001.

[18] Seymour K, A YarKhan, S Agrawal, J. Dongarra, "NetSolve: Grid Enabling Scientific Computing Environments*", Grid Computing and New Frontiers of High Performance Processing*, vol. 14, pp. 33-51, Netherlands, 2005.

[19] Subodha Kumar, Kaushik Dutta, Vijay Mookerjee, " Maximizing business value by optimal assignment of jobs to resources in grid computing", *European journal of scientific research,* Vol. 194, Issue 3, pp. *856-872, April 2009.*

[20] Vadhiyar S, Dongarra J, Yarkhan A, "GrADSolve - RPC for High Performance Computing on the Grid", *in the proceedings of 9th International Euro-Par Conference,* Springer, LCNS 2790, pp. 394-403, August 26 -29, 2003.

[21] Vijay, Chuang Liu K. Naik, Chuang Liu, Jonathan Wagner, "On-line Resource Matching for Heterogeneous Grid Environments", *IEEE International Symposium on Cluster Computing and the Grid,* Vol. 2, pp. 607 – 614, 2005.

[22] Vladimir V. Korkhova, Jakub T. Moscicki, Valeria V. Krzhizhanovskaya, "Dynamic workload balancing of parallel applications with user-level scheduling on the Grid", *Future Generation Computer Systems,* Vol. 25, pp 268-34, 2009.